UNIVERSITY OF CAMBRIDGE

Multimodal learning and language models for enhanced knowledge representations

Adrián Rodríguez-Bazaga

Fitzwilliam College

This dissertation is submitted on April, 2025
for the degree of Doctor of Philosophy

DECLARATION

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the preface and specified in the text. It is not substantially the same as any work that has already been submitted, or is being concurrently submitted, for any degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Adrián Rodríguez-Bazaga
April, 2025

ABSTRACT

Multimodal learning and language models for enhanced knowledge representations

Adrián Rodríguez-Bazaga

Machine learning with modern neural architectures, particularly large-scale language models, has enabled impressive progress across diverse problem domains such as natural language processing, graph representation learning, and tabular data analysis. However, these successes have predominantly relied on abundant, single-modal datasets, often overlooking the inherently multimodal and structurally complex nature of real-world data. Real-world data typically combines multiple modalities—text, graph structures, and tabular formats—each encoding distinct but complementary insights. Effectively integrating these heterogeneous sources remains one of the most significant open challenges in artificial intelligence research.

In this thesis, I directly address this challenge by proposing three pioneering contributions that leverage multimodal data through sophisticated neural architectures and advanced representation learning techniques. First, I introduce HyperBERT, a novel neural architecture that integrates hypergraph neural networks with pretrained language models, achieving superior node classification performance on text-attributed hypergraphs. The second contribution tackles dense retrieval in environments with limited labeled data by developing a novel weakly supervised semantic distillation framework. Leveraging the rich semantic understanding embedded in large language models along with negative sampling strategies, this framework significantly improves retrieval accuracy and generalizability, particularly for tasks such as claim verification and evidence retrieval. The third contribution introduces TabMDA, a Transformer-based manifold data augmentation method for tabular data, leveraging pretrained Transformer encoders in combination with an advanced in-context subsetting technique. This approach generates synthetic samples that faithfully preserve data dependencies, enhancing downstream classification performance.

Collectively, these interconnected contributions significantly advance the frontier of multimodal representation learning. Through the explicit modeling and integration of structural, textual, and tabular information, this thesis paves the way towards building more sophisticated, adaptable, and intelligent computational systems capable of nuanced understanding and interaction with complex real-world information structures.

ACKNOWLEDGEMENTS

Completing this thesis marks the end of an intense and rewarding six-year journey, one which tested my resilience and shaped me profoundly, both professionally and personally. Undertaking a self-funded PhD alongside full-time employment was incredibly challenging in ways I could not have anticipated, yet it provided invaluable lessons and growth opportunities for which I am immensely grateful.

First and foremost, my deepest gratitude goes to my parents: Pipo and Yolanda. Coming from a low-income background, my parents provided unwavering emotional support, encouragement, and love despite limited resources. Their sacrifices enabled me to pursue my academic dreams, even in the face of substantial financial constraints. This thesis is dedicated to their relentless perseverance, generosity, and belief in my potential. Thank you, from the bottom of my heart.

I want to extend heartfelt thanks to my supervisors, Pietro Liò (Computer Science and Technology) and Gos Micklem (Genetics). Gos was incredibly supportive since the very start, displaying remarkable flexibility and patience as my research gradually evolved beyond our initial plans. Interestingly, it was one of our very early conversations, exploring how machine learning could automate database querying, that unintentionally sparked my lasting fascination with language modelling, a domain that has defined my academic and professional journey ever since. Pietro's arrival as my supervisor in 2021 marked a pivotal shift in my research towards deep learning and Transformer-based methods. His mentorship, enthusiasm, and guidance allowed me to immerse myself deeply in these topics, significantly enriching the impact and quality of my work. I am deeply grateful to both of them for their extraordinary understanding and support, especially given the unique challenges posed by balancing PhD research with professional commitments.

My professional and personal growth were immensely influenced by internships at Microsoft Research in Cambridge and Amazon Science AGI in Berlin. These opportunities were transformative experiences, providing exposure to cutting-edge research environments and allowing me to collaborate with inspiring mentors and colleagues. Both internships profoundly shaped my skills and perspectives, solidifying my passion for machine learning and preparing me for a research career.

Special thanks go to Microsoft, where I am now privileged to work full-time on topics directly aligned with my PhD research. Being able to professionally pursue my greatest interests

represents a fulfilling culmination of this challenging yet enriching journey.

Finally, I extend my sincere appreciation to friends and colleagues who supported me along this path. Your companionship, encouragement, and advice made this challenging experience more manageable and enjoyable. Each of you contributed uniquely to my growth, and for that, I am deeply thankful.

To everyone who supported me along this extraordinary journey, I express my deepest gratitude, this achievement would not have been possible without you: Thank you!

CONTENTS

1 Introduction **15**

1.1 Motivation and research questions 16

1.2 Thesis structure 18

1.3 List of publications 20

1.3.1 Main publications 20

1.3.2 Related publications 21

2 Background **23**

2.1 Fundamentals of deep neural networks 23

2.1.1 Feedforward neural networks 23

2.1.1.1 The perceptron 23

2.1.1.2 Multilayer perceptrons 24

2.1.1.3 Learning process 24

2.1.2 Neural networks for sequences 28

2.1.2.1 Recurrent neural networks 28

2.1.2.2 Long short-term memory networks 29

2.1.3 Neural networks for graphs 31

2.1.3.1 Graph neural networks 32

2.1.3.2 Graph convolutional networks 32

2.1.3.3 Graph attention networks 33

2.2 Modern architectures for language modelling 34

2.2.1 The Transformer 34

2.2.1.1 Self-attention mechanism 35

2.2.1.2 Positional encodings 36

2.2.1.3 Encoder and decoder 36

2.2.1.4 Encoder-only Transformer 37

2.2.1.5 Decoder-only Transformer 38

2.2.1.6 Encoder-decoder Transformer 38

2.2.2 Large language models 38

2.2.2.1 From static to contextual representations 39

2.2.2.2 Pre-training and fine-tuning paradigm . . . 39
2.2.2.3 Architectural innovations and their impact . . . 39
2.3 Multimodal learning in large language models . . . 40
2.3.1 Beyond single-modal learning . . . 40
2.3.2 Architectural foundations of multimodal language models . . . 40
2.3.2.1 Modality-specific encoders . . . 40
2.3.2.2 Cross-modal fusion mechanisms . . . 41
2.3.2.3 Unified multimodal Transformer architectures . . . 41
2.4 Machine learning tasks . . . 41
2.4.1 Text-attributed node classification on hypergraphs . . . 41
2.4.1.1 Problem formulation . . . 42
2.4.1.2 Approach overview . . . 42
2.4.2 Semantic matching for dense retrieval . . . 42
2.4.2.1 Problem formulation . . . 43
2.4.2.2 Application to claim verification . . . 44
2.4.3 Tabular data augmentation with pretrained Transformers . . . 44
2.4.3.1 Problem formulation . . . 44
2.4.3.2 Transformer-based manifold data augmentation . . . 45
2.4.3.3 Application to downstream tasks . . . 45

3 Higher-order representations for text-attributed node classification 47
3.1 Introduction and contribution overview . . . 47
3.2 Related work . . . 48
3.3 Preliminaries . . . 50
3.3.1 Text-attributed hypergraphs . . . 50
3.3.2 Pretrained language models for node classification . . . 51
3.3.3 Problem formulation . . . 51
3.4 Methodology . . . 52
3.4.1 The HyperBERT layer . . . 52
3.4.2 Hypergraph-aware pretraining task . . . 54
3.4.3 Fine-tuning in downstream tasks . . . 56
3.5 Experiments . . . 57
3.5.1 Datasets and evaluation metrics . . . 57
3.5.2 Implementation details . . . 57
3.5.3 Overall performance . . . 58
3.5.4 Ablation studies . . . 58
3.6 Discussion and summary . . . 60

4 Weakly supervised language model semantic distillation for dense retrieval 63
4.1 Introduction and contribution overview . . . 63
4.2 Related work . . . 64
4.3 Methodology . . . 66
4.3.1 Data processing pipeline . . . 67
4.3.2 Pretraining method . . . 67
4.3.3 Generation of negative instances . . . 69
4.3.4 Language model distillation for semantic matching . . . 70
4.4 Experiments . . . 71
4.4.1 Implementation details . . . 71
4.4.2 Experiments on the FEVER dataset . . . 73
4.4.3 Experiments on the FB15k-237 dataset . . . 75
4.4.4 Ablation studies . . . 75
4.5 Discussion and summary . . . 77

5 Tabular manifold data augmentation leveraging pre-trained Transformers 79
5.1 Introduction and contribution overview . . . 79
5.2 Related work . . . 81
5.3 Preliminaries . . . 82
5.3.1 Problem formulation . . . 82
5.3.2 Pretrained Transformers for tabular data . . . 83
5.4 Methodology . . . 83
5.4.1 Embedding tabular data into the manifold space . . . 83
5.4.2 Manifold data augmentation with in-context subsetting . . . 84
5.4.3 Training downstream classifiers on augmented data . . . 84
5.5 Experiments . . . 85
5.5.1 Experimental setup . . . 85
5.5.2 Overall performance on classification accuracy . . . 87
5.5.3 Effect of in-context subsetting on classification performance . . . 89
5.5.4 Qualitative analysis of original data versus generated manifold space . . . 91
5.6 Discussion and summary . . . 93

6 Conclusions and future directions 95

Bibliography 97

GLOSSARY

Batch Normalisation A method for faster and more stable training of neural networks by normalising intermediate representations.

CNN Convolutional Neural Network, a neural architecture leveraging spatial inductive biases and convolution operations, typically used in image-related tasks.

Dense Retrieval Information retrieval techniques leveraging neural embeddings of text to identify relevant documents or data points from large datasets.

GAT Graph Attention Network, a subtype of GNN utilizing attention mechanisms to weigh the importance of neighboring nodes dynamically.

GCN Graph Convolutional Network, a subtype of GNN employing convolution operations adapted for graph-structured data.

GNN Graph Neural Network, a type of neural network incorporating structural biases by utilizing connectivity information in the input graph.

Hypergraph A generalized graph structure where edges, known as hyperedges, can connect more than two nodes, capturing higher-order relationships.

LLM Large Language Model, a type of deep learning model designed to understand and generate human language, trained on vast textual datasets.

LSTM Long Short-Term Memory, a type of recurrent neural network designed to effectively learn long-range dependencies in sequential data.

MLP Multilayer perceptron, a type of neural network consisting of multiple fully connected layers.

MP Message Passing, a paradigm used in graph neural networks for aggregating information across graph-structured data by exchanging messages between nodes.

Semantic Distillation A technique that transfers semantic knowledge from large language models to smaller models or downstream tasks, enhancing their semantic understanding.

SGD Stochastic Gradient Descent, a common optimization method used for training neural networks through iterative parameter updates.

Transformer A deep learning model architecture characterized by self-attention mechanisms, widely used in NLP tasks and foundational to modern large language models.

CHAPTER 1

INTRODUCTION

The field of machine learning has undergone transformative advancements in recent years, driven primarily by breakthroughs in deep neural network architectures and the widespread availability of large-scale data. Initial developments in feedforward and recurrent architectures laid the groundwork for modern models, yet these early networks often struggled to capture long-range dependencies and intricate interactions present in real-world data. The emergence of Transformer architectures [1] and their subsequent evolution into large language models (LLMs) [2, 3] have fundamentally reshaped how sequential and contextual information is represented, enabling unprecedented advances in natural language processing and setting the stage for integrating diverse modalities within unified machine learning frameworks.

In parallel, research in graph representation learning has seen significant advances through Graph Neural Networks (GNNs), including Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs) [4, 5]. Nevertheless, real-world graphs frequently come coupled with high-dimensional, unstructured modalities (particularly text) that require more nuanced representation learning strategies. Hypergraphs, capable of modeling higher-order relationships among multiple entities, introduce additional complexity and representational potential, motivating methods that integrate structural connectivity with rich textual semantics. Such integration is essential for tasks like node classification, where both local neighborhood structure and broader semantic context significantly influence predictive outcomes.

Concurrently, several machine learning domains have increasingly embraced weakly supervised strategies to overcome limitations posed by reliance on explicit annotations, which can be costly and scarce. Using the semantic and representational capabilities of pretrained language models, recent methods [6, 7] demonstrate the effectiveness of weakly supervised approaches across various tasks. These methods exploit sophisticated embeddings learned by Transformer-based architectures, enabling robust learning and high accuracy even in environments with minimal or indirect supervision. This shift exemplifies how modern machine learning frameworks leverage pretrained representations to achieve significant performance improvements

without extensive labeled datasets, enabling opportunities for harnessing such capabilities in typically data-scarce domains such as information retrieval.

Another rapidly evolving research area is the application of pretrained Transformers to tabular data, traditionally dominated by classical machine learning techniques such as tree-based models. Recent developments reveal that Transformer models effectively capture the complex interdependencies inherent to tabular data by embedding inputs into continuous latent manifolds [8, 9]. This property opens opportunities for innovative data augmentation approaches directly within these embedding spaces, particularly valuable for enhancing downstream classifier performance on challenging, heterogeneous datasets.

This thesis introduces interconnected research contributions that advance multimodal learning by exploiting pretrained language models as powerful underlying foundations for sophisticated representation learning. Specifically, I introduce HyperBERT, an innovative method for node classification in text-attributed hypergraphs that effectively integrates higher-order structural and textual inductive biases into hypergraph-based neural architectures and language models. Then I propose SFAVEL, a robust weakly supervised dense retrieval approach, designed for data-scarce scenarios, which leverages semantic distillation from pretrained language models and negative sampling strategies for claim–evidence matching. Lastly, I present TabMDA, a training-free tabular data augmentation method that leverages the latent manifold embedding spaces learned by pretrained tabular Transformer models. By developing the novel in-context subsetting (ICS) mechanism, TabMDA generates diverse and informative augmented representations, significantly improving downstream classifier training performance. Collectively, these contributions demonstrate the wide-ranging applicability and effectiveness of pretrained models and their learned embedding spaces across various tasks, leveraging structural, textual, and tabular data modalities for enhanced knowledge representations.

In essence, this thesis contributes to the emerging field of multimodal representation learning by addressing critical limitations inherent to traditional single-modal machine learning approaches. By proposing novel methods that integrate pretrained language models with hypergraph neural networks, weakly supervised dense retrieval techniques, and latent tabular data augmentation techniques, the research presented herein substantially advances state-of-the-art methodologies. Collectively, these contributions establish foundational principles for more versatile, robust, and effective analyses of complex real-world information structures, opening promising avenues for future exploration in machine learning research.

1.1 Motivation and research questions

This thesis proposes a set of theoretical contributions in the form of machine learning methodologies that advance three key areas: text-attributed node classification in hypergraphs, weakly supervised dense retrieval through semantic distillation, and tabular data augmentation by lever-

aging pretrained Transformer models. These contributions are unified by their common strategy of exploiting pretrained models and their learned latent embedding spaces, enabling sophisticated representation learning across complex data structures, while preserving practical applicability and performance improvements in downstream tasks.

The research was motivated by several key observations about current limitations in the field. First, while graph neural networks have shown remarkable success in many domains, they often struggle to effectively incorporate rich textual node attributes, particularly in hypergraph settings where relationships extend beyond pairwise connections. Second, existing approaches to dense retrieval frequently rely on supervised learning, limiting their applicability in scenarios where labeled data is scarce. Third, the challenge of data augmentation for tabular domains remains understudied compared to computer vision and natural language processing, despite its critical importance for real-world applications.

Within these areas, this thesis addresses three central research questions:

1. *How can we effectively leverage both structural and textual information in hypergraph-based learning?* While recent advances in language models have revolutionized text processing, integrating these capabilities with graph neural networks remains challenging, particularly for hypergraphs where relationships are more complex than traditional graphs. Chapter 3 addresses this through the development of higher-order representations for text-attributed node classification, proposing a novel language model architecture that can effectively process both hypergraph structures and textual content.

2. *Can we develop robust weakly supervised approaches for dense retrieval that leverage the semantic capabilities of large language models?* Traditional dense retrieval systems often require substantial labeled data for training, limiting their applicability. Chapter 4 explores this challenge by developing a novel weakly supervised semantic distillation technique that leverages language models, enabling effective dense retrieval without the need for vast supervised training data.

3. *How can we leverage pre-trained Transformer encoders to enable effective data augmentation for tabular domains?* While Transformers have achieved remarkable success in text and image domains, their application to the tabular domain presents unique challenges. Chapter 5 investigates this question by developing a novel Transformer-based data augmentation technique specifically designed to improve performance on tabular data tasks, addressing the particular characteristics and constraints of this domain.

These research questions are explored through a combination of theoretical development and empirical validation. Each contribution is evaluated on challenging real-world datasets and compared against state-of-the-art baselines. The thesis not only proposes novel methodologies but also provides insights into the fundamental limitations and trade-offs inherent in each

approach. The methodological contributions span multiple aspects of machine learning, from architectural innovations in neural networks to novel applications of language model distillation and Transformer-based data augmentation. Throughout the thesis, particular attention is paid to both theoretical soundness and practical applicability, ensuring that the proposed solutions can be effectively deployed in real-world scenarios. Figure 1.1 provides a condensed overview of this thesis.

1.2 Thesis structure

The rest of this thesis is structured as follows: contribution-wise, I first tackle text-attributed node classification with higher-order representations, followed by weakly supervised dense retrieval using language model semantic distillation, and conclude with tabular data augmentation leveraging pre-trained Transformers. Each chapter builds upon the theoretical foundations laid out in the background while addressing distinct challenges in machine learning.

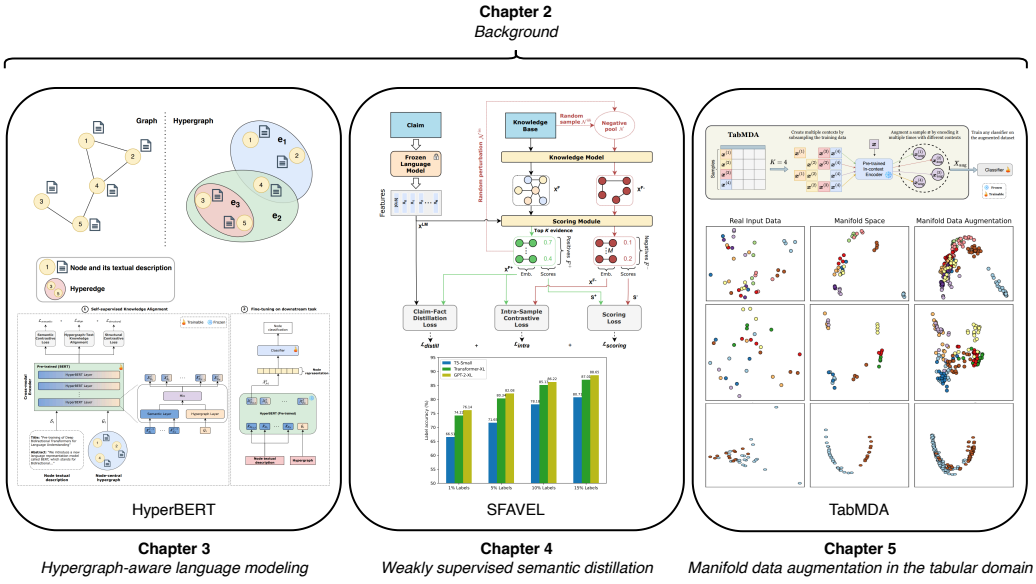

Figure 1.1: An overview of the main contributions presented in this dissertation. Firstly, a novel methodology (HyperBERT) is introduced for text-attributed node classification in hypergraphs, effectively integrating structural and textual inductive biases by leveraging pretrained language models within hypergraph-based neural architectures (Chapter 3). Secondly, the SFAVEL framework is proposed, providing a weakly supervised dense retrieval approach that exploits rich semantic inductive biases distilled from pretrained language models, particularly addressing challenges associated with limited labeled data through innovative claim-evidence semantic matching and negative instance generation (Chapter 4). Lastly, a novel training-free tabular manifold data augmentation method (TabMDA) is developed, leveraging the learned latent manifold of pretrained tabular Transformer models. By introducing the in-context subsetting (ICS) mechanism, TabMDA generates diverse and informative augmented representations directly in the embedding space, significantly boosting the performance of downstream classifiers (Chapter 5). Collectively, these contributions share the common strategy of harnessing pretrained Transformer-based models and their learned embedding spaces, demonstrating their broad applicability and effectiveness in solving diverse machine learning challenges across structural, textual, and tabular data modalities.

Chapter 2: Background. This chapter lays the theoretical foundation necessary for understanding the research presented in the contribution chapters. The chapter begins with fundamentals of deep neural networks, covering feedforward architectures, the perceptron, multilayer perceptrons, and the learning process. It then progresses to neural networks for sequences, examining recurrent neural networks, long short-term memory networks, as well as neural networks for graphs, including graph neural networks, graph convolutional networks, and graph attention networks. Then, I cover modern architectures for language models, focusing on the Transformer architecture, and define the concept of ”large language model”. The chapter concludes with an overview of multimodal learning in LLMs, contrasting single- versus multi-modal scenarios, and examining the architectural foundations for such models.

Chapter 3: Higher-order representations for text-attributed node classification. This chapter introduces novel approaches for leveraging both textual and structural information in hypergraph-based learning. After establishing the theoretical context and related work, the chapter presents the concept of Text-Attributed Hypergraphs (TAHG) and discusses how pretrained language models can be adapted for node classification in such setting. The methodology centers on the HyperBERT layer and introduces a hypergraph-aware pretraining task, followed by fine-tuning strategies for downstream applications. Comprehensive experiments demonstrate the effectiveness of the approach across multiple datasets and evaluation metrics.

Chapter 4: Weakly supervised dense retrieval through language model semantic distillation. This chapter introduces an innovative framework for dense retrieval that leverages semantic distillation from pretrained language models in a weakly supervised manner. Rather than depending heavily on labeled data, the approach exploits the rich semantic representations of large language models to guide retrieval tasks, particularly in contexts where annotated datasets are limited. A key contribution is the introduction of a language model-based claim-evidence distillation technique and novel negative instance generation strategies, which together significantly improve dense retrieval performance. Comprehensive experiments and ablation studies validate the robustness and efficacy of this method across diverse datasets and retrieval scenarios.

Chapter 5: Tabular manifold data augmentation leveraging pre-trained Transformers. This chapter addresses the challenge of improving machine learning performance on tabular tasks by introducing a novel training-free data augmentation method, termed TabMDA. Instead of developing new model architectures, my approach leverages the latent manifold learned by pretrained tabular Transformer models to generate diverse augmented data directly in the embedding space. I propose a novel in-context subsetting (ICS) mechanism, which creates meaningful label-invariant augmentations, significantly enhancing the performance and robustness of various

downstream classifiers. Extensive experiments demonstrate notable accuracy gains across multiple tabular datasets, confirming the effectiveness and practicality of the approach.

Chapter 6: Conclusion and future directions. Finally, this chapter synthesises the key contributions presented throughout the thesis and outlines promising directions for future research. It reflects on how the developed methods advance our understanding of machine learning across three key areas: text-attributed node classification, weakly supervised dense retrieval, and tabular data augmentation. The chapter concludes by discussing potential extensions and applications of the proposed approaches.

Each contribution chapter (3-5) follows a consistent structure: beginning with an introduction and contribution overview, followed by related work, methodology development, experimental validation, and concluding with discussion and summary. To ensure reproducibility and transparency of the presented methods, all code implementations described in the thesis are publicly available on GitHub. Specifically, the implementation of Chapter 3 is accessible at `https://github.com/AdrianBZG/HyperBERT`, Chapter 4 at `https://github.com/AdrianBZG/SFAVEL`, and Chapter 5 at `https://github.com/AdrianBZG/TabMDA`.

1.3 List of publications

The research efforts presented in this thesis have led to several publications, which I enumerate below. The main three publications form the core chapters of this thesis, while the remaining ones represent closely or parallel related work that has influenced or emerged from the main research direction.

1.3.1 Main publications

1. **Bazaga, A.**, Lio, P., & Micklem, G. (2024). *HyperBERT: Mixing Hypergraph-Aware Layers with Language Models for Node Classification on Text-Attributed Hypergraphs*. Conference on Empirical Methods in Natural Language Processing (EMNLP 2024), pages 9181–9193. Association for Computational Linguistics.

   This work, presented in Chapter 3, introduces a novel approach for leveraging both textual and structural information in hypergraph-based tasks. The paper proposes an architecture that effectively combines hypergraph neural networks with pre-trained language models, achieving state-of-the-art results in text-attributed node classification tasks.

2. **Bazaga, A.**, Lio, P., & Micklem, G. (2024). *Unsupervised Pretraining for Fact Verification by Language Model Distillation*. The Twelfth International Conference on Learning Representations (ICLR 2024).

This publication, forming Chapter 4, presents a novel framework for weakly supervised language model distillation for dense retrieval, applied to the task of claim verification. In particular, it addresses the challenge of aligning textual claims with evidence without relying on vast amounts of annotated corpora. Instead of solely relying on labelled data, I leverage pre-trained language models to distil self-supervised features that are both semantically rich and compact for claim-evidence matching. A novel multi-term contrastive loss function is introduced to ensure robust claim–evidence alignment by preserving high-quality semantic relationships across the corpus.

3. **Bazaga, A.**, Margeloiu, A., Simidjievski, N., Lio, P., & Jamnik, M. (2024). *TabMDA: Tabular Manifold Data Augmentation for Any Classifier using Transformers with In-context Subsetting*. International Conference on Machine Learning (ICML 2024).

   This work, detailed in Chapter 5, introduces a novel approach to tabular data augmentation using Transformer architectures. The method leverages in-context learning to generate synthetic samples in the latent domain that preserve the underlying data distribution, significantly improving classification performance of downstream classifiers across diverse datasets.

1.3.2 Related publications

The following publications, while not forming core chapters of the thesis, represent significant parallel work that has either influenced or emerged from the main research directions:

1. **Bazaga, A.**, Lio, P., & Micklem, G. (2024). *Language Model Knowledge Distillation for Efficient Question Answering in Spanish*. The Twelfth International Conference on Learning Representations (ICLR 2024).

   This publication leverages knowledge distillation techniques to the domain of low-resource multilingual question answering, specifically focusing on Spanish language understanding. The work demonstrates the broader applicability of distillation approaches beyond dense retrieval.

2. Zhu, M., **Bazaga, A.**, & Liò, P. (2024). *FLUID-LLM: Learning Computational Fluid Dynamics with Spatiotemporal-aware Large Language Models*. arXiv preprint arXiv:2406.04501.

   This work explores the application of large language models to computational fluid dynamics, developing a novel architecture that incorporates spatio-temporal awareness using graph neural networks. The research demonstrates how complex physical systems can be effectively modelled using multimodal neural architectures, sharing methodological similarities with the work presented in Chapter 3.

3. **Bazaga, A.**, Lio, P., & Micklem, G. (2023). *SQLformer: Deep Auto-Regressive Query Graph Generation for Text-to-SQL Translation*. arXiv preprint arXiv:2310.18376.

   This research introduces a novel Transformer-based architecture for text-to-SQL translation, incorporating techniques for structural information processing related to the approaches developed in Chapter 3. The work demonstrates state-of-the-art performance across multiple text-to-SQL benchmarks.

4. **Bazaga, A.**, Gunwant, N., & Micklem, G. (2021). *Translating synthetic natural language to database queries with a polyglot deep learning framework*. Scientific Reports, 11, 18462.

   This early publication introduces a framework for translating natural language to database queries, significantly influencing my subsequent works on language modelling. Moreover, it was instrumental in shaping my research trajectory, drawing me towards advanced language modelling techniques and inspiring my deeper exploration into Transformer-based methods during the remainder of my doctoral studies.

All these publications have undergone peer review and most have been presented at leading international conferences or published in top-tier journals, demonstrating the broad impact and academic rigour of the research carried out as part of this thesis. The work spans multiple areas of machine learning, from language models to graph neural networks and data augmentation, while maintaining a coherent focus on advancing the state-of-the-art in structured data processing and multi-modal knowledge representations. Any unpublished work has been a strategic decision throughout my PhD to maximize focus on the most impactful contributions.

CHAPTER 2

BACKGROUND

This dissertation's contributions, as detailed in Section 1.3, are solely focused on novel architectures and systems based on deep neural networks. To ensure a thorough understanding and interpretation of their implications, I will present a comprehensive overview of the most crucial foundational methods in this chapter.

2.1 Fundamentals of deep neural networks

2.1.1 Feedforward neural networks

Feedforward neural networks are a class of machine learning models that consist of layers of interconnected processing units, commonly referred to as neurons or perceptrons [10]. These networks propagate information in a single direction, from the input to the output, without forming cycles, making them one of the simplest yet most powerful architectures in machine learning.

2.1.1.1 The perceptron

The basic building block of feedforward networks is the perceptron. A perceptron receives an input vector $\vec{x} \in \mathbb{R}^n$ and computes a weighted sum of its components, augmented by a bias term. Formally, given a weight vector $\vec{w} \in \mathbb{R}^n$ and a bias b, the perceptron computes

$$z = b + \vec{w}^T \vec{x}.$$

This linear combination is then typically passed through a nonlinear activation function σ, yielding the final output:

$$y = \sigma(z) = \sigma\left(b + \vec{w}^T \vec{x}\right).$$

Several activation functions are commonly employed in practice. For example, the logistic sigmoid function, defined as $\sigma(x) = \frac{1}{1+e^{-x}}$, maps any real number into the interval $[0, 1]$, making it useful for modeling probabilities in binary classification tasks. Alternatively, the hyperbolic tangent function, given by $\sigma(x) = \tanh(x)$, outputs values in the range $[-1, 1]$ and is often preferred when centered, real-valued outputs are desired. Another popular choice is the rectified linear unit (ReLU), defined as $\sigma(x) = \max(0, x)$, which is favored for its simplicity, computational efficiency, and its ability to mitigate the vanishing gradient problem. A variant of ReLU, known as Leaky ReLU, is defined piecewise such that $\sigma(x) = x$ for $x \geq 0$ and $\sigma(x) = \alpha x$ for $x < 0$, where α is a small positive constant that ensures the gradient remains non-zero for negative inputs.

2.1.1.2 Multilayer perceptrons

Building upon the single perceptron, multilayer perceptrons (MLPs) stack several layers of neurons in a feedforward fashion, allowing the network to learn more complex, hierarchical representations. In an MLP, each neuron in a layer receives inputs from all neurons in the preceding layer. For a given layer with input $\vec{x}$, the output of the j-th neuron is computed as

$$y_j = \sigma\left(b_j + \vec{w}_j^T \vec{x}\right), \tag{2.1}$$

where $\vec{w}_j$ is the weight vector and b_j is the bias associated with the neuron.

For convenience and computational efficiency, the operations of an entire layer are often expressed in matrix form:

$$\vec{y} = \sigma\left(W\vec{x} + \vec{b}\right), \tag{2.2}$$

with W being the weight matrix and $\vec{b}$ the bias vector for that layer. Stacking multiple layers results in deeper architectures:

$$\vec{y} = \sigma_2\left(W_2\,\sigma_1\left(W_1\vec{x} + \vec{b}_1\right) + \vec{b}_2\right). \tag{2.3}$$

While the universal approximation theorem [11] guarantees that a single hidden layer with sigmoidal activations can approximate any bounded continuous function to arbitrary precision, this theoretical result does not provide practical guidelines for network design. In practice, deeper networks are preferred as they enable the hierarchical extraction of features, thereby simplifying the representation of complex functions despite the potential increase in the number of parameters.

2.1.1.3 Learning process

In a fixed multilayer perceptron (MLP), where the architecture (i.e. the number of neurons, the structure of weight matrices W_i, bias vectors $\vec{b}_i$, and the forms of the activation functions σ_i) is

predetermined, the behavior of the network is entirely specified by the values of these parameters. The learning process involves adjusting these parameters incrementally from an initial setting so as to minimize a task-specific loss function.

The initialisation of the network parameters is crucial, as it can significantly impact the convergence and overall performance of the model. Early deep learning approaches often relied on unsupervised pretraining methods such as deep belief networks [12] or stacked autoencoders [13, 14]. Nowadays, however, it is standard practice to initialise weights by drawing each one independently from a carefully chosen distribution, typically either a uniform or a normal distribution with zero mean. Only the variance of the weights, denoted σ^2, must then be specified. For a linear neuron with zero-mean, uncorrelated inputs $\vec{x}$ and weights $\vec{w}$, the variance of its output is given by:

$$\mathrm{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = n_{in}\,\sigma^2\,\mathrm{Var}(X), \tag{2.4}$$

where n_{in} is the number of inputs. To preserve the input variance through the layer, an important consideration for stable gradient propagation, it is common to set $\sigma^2 = \frac{1}{n_{in}}$, which is known as LeCun initialisation [15]. Alternatively, to preserve the variance of the backward-propagated gradients, one may require $\sigma^2 = \frac{1}{n_{out}}$, where n_{out} is the number of output connections. Balancing these two criteria leads to Xavier initialisation [16]:

$$\sigma^2 = \frac{2}{n_{in} + n_{out}}, \tag{2.5}$$

which is particularly effective for neurons with roughly linear activations in their unsaturated regime. For ReLU-like activations, where only positive activations are passed through, He initialisation [17] is preferred:

$$\sigma^2 = \frac{2}{n_{in}}, \tag{2.6}$$

a scheme that has been instrumental in achieving strong performance on large-scale benchmarks.

Once the parameters are initialised, the network is trained by minimizing a differentiable loss function $\mathcal{L}$, which quantifies the discrepancy between the network's predictions and the desired outputs. In supervised learning, for example, we assume a training set $\mathcal{S} = \{(\vec{x}_i, \vec{y}_i)\}_{i=1}^{m}$. For regression tasks, a common choice is the mean squared error:

$$\mathcal{L}_{MSE} = \frac{1}{m}\sum_{i=1}^{m} \|\vec{y}_i - \hat{\vec{y}}_i\|^2, \tag{2.7}$$

where $\hat{\vec{y}}_i$ is the network's output for input $\vec{x}_i$. For classification tasks, the network's raw outputs $\vec{z}_i$ are first converted into probabilities via the softmax function:

$$\hat{y}_{ij} = \frac{\exp(z_{ij})}{\sum_{k=1}^{K} \exp(z_{ik})}, \tag{2.8}$$

and the loss is then measured using cross-entropy:

$$\mathcal{L}_{CE} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{K} y_{ij}\log \hat{y}_{ij}. \tag{2.9}$$

The parameters are updated using mini-batch stochastic gradient descent (SGD). At each iteration, a mini-batch $\mathcal{B}$ is sampled from the training set, and the gradient of the loss with respect to the parameters, $\nabla_{\vec{\theta}}\mathcal{L}_{\mathcal{B}}$, is computed via backpropagation [18]. A typical update rule is:

$$\vec{\theta}_{t+1} \leftarrow \vec{\theta}_t - \eta \nabla_{\vec{\theta}}\mathcal{L}_{\mathcal{B}}\Big|_{\vec{\theta}=\vec{\theta}_t}, \tag{2.10}$$

where η is the learning rate. The choice of learning rate is critical; adaptive optimisation algorithms such as Adam [19] have become popular because they adjust the learning rate dynamically based on historical gradient information.

In Adam, for each parameter, the gradient $\vec{g}_t$ is computed at time step t, and exponential moving averages of the gradients and their squares are maintained:

$$\vec{m}_{t+1} = \beta_1 \vec{m}_t + (1-\beta_1)\vec{g}_t, \tag{2.11}$$

$$\vec{v}_{t+1} = \beta_2 \vec{v}_t + (1-\beta_2)\vec{g}_t^2, \tag{2.12}$$

with default decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Bias corrections are then applied:

$$\hat{\vec{m}}_t = \frac{\vec{m}_t}{1-\beta_1^t}, \tag{2.13}$$

$$\hat{\vec{v}}_t = \frac{\vec{v}_t}{1-\beta_2^t}, \tag{2.14}$$

followed by the parameter update:

$$\vec{\theta}_{t+1} \leftarrow \vec{\theta}_t - \eta \frac{\hat{\vec{m}}_t}{\sqrt{\hat{\vec{v}}_t} + \epsilon}, \tag{2.15}$$

where ϵ is a small constant (typically 10^{-8}) that prevents division by zero.

To mitigate overfitting, an especially pressing issue in deep networks, which are typically overparameterized relative to the size of the training data, regularisation techniques are employed. A common approach is L_2 regularisation, which adds a penalty proportional to the square of the weights' magnitudes:

$$\tilde{\mathcal{L}} = \mathcal{L} + \frac{\lambda}{2}\|\vec{\theta}\|^2, \tag{2.16}$$

where λ controls the trade-off between fitting the training data and keeping the weights small.

In practice, training proceeds by shuffling the dataset and processing it in mini-batches, iterating over the entire set (one epoch) and repeating the process until convergence. This

combination of careful initialisation, adaptive optimisation, and regularisation enables deep neural networks to learn complex functions while maintaining generalisability to unseen data.

Dropout. Neural networks trained without additional regularisation tend to develop highly specialised neurons, sometimes even relying on certain neurons to correct the errors of others. This over-specialisation can result in a network that is overly dependent on individual neurons, so that the failure or poor performance of a single neuron might degrade the overall network performance. The dropout technique [20] addresses this issue by randomly "dropping", or setting to zero, the output of each neuron with a fixed probability p during training. To maintain the overall scale of the activations, the outputs of the remaining neurons are then scaled by a factor of $\frac{1}{1-p}$. This stochastic removal of neurons forces the network to develop redundant representations, thereby reducing overreliance on any single neuron and improving generalisation.

Batch Normalisation. During training, the distribution of inputs to each layer in a deep network can shift as the parameters of previous layers are updated, a phenomenon known as internal covariate shift. While weight initialisation strategies help preserve input statistics at the outset, these statistics can drift over the course of training, complicating the learning process. Batch normalisation [21] offers an effective remedy by normalising the activations of each layer on a per-mini-batch basis. Given a mini-batch $B = \{\vec{x}_1, \ldots, \vec{x}_m\}$, the method first computes the mini-batch mean

$$\vec{\mu}_B = \frac{1}{m} \sum_{i=1}^{m} \vec{x}_i,$$

and the mini-batch variance

$$\vec{\sigma}_B^2 = \frac{1}{m} \sum_{i=1}^{m} (\vec{x}_i - \vec{\mu}_B)^2.$$

Each activation is then normalised as

$$\hat{\vec{x}}_i = \frac{\vec{x}_i - \vec{\mu}_B}{\sqrt{\vec{\sigma}_B^2 + \epsilon}},$$

and subsequently scaled and shifted by learnable parameters
γ and β:

$$\hat{\vec{y}}_i = \gamma \, \hat{\vec{x}}_i + \beta.$$

This procedure not only stabilizes and accelerates training but also allows the network to learn the optimal scale and shift for the normalised activations. During inference, since mini-batches are no longer available, the normalisation is performed using estimates of the mean and variance computed over the entire training set, typically maintained as an exponential moving average

during training.

2.1.2 Neural networks for sequences

While multilayer perceptrons excel at processing flat, unstructured data, many real-world applications, such as natural language and speech processing, exhibit inherent sequential structure. Exploiting this structure allows models to incorporate useful inductive biases, thereby reducing the amount of training data required. In this section, I discuss neural architectures designed specifically for sequential data, noting that although these models can be framed as constrained MLPs, their explicit treatment of sequence order enables much more efficient learning.

2.1.2.1 Recurrent neural networks

For sequences where inputs arrive over an arbitrary number of time steps with fixed feature dimensions, fully-connected layers are inadequate due to their fixed-size input requirements. Recurrent Neural Networks (RNNs) address this limitation by introducing a mechanism that iteratively processes the sequence while maintaining a summary of past information. This summary is updated at each time step to reflect both the current input and the accumulated context from previous steps.

The Recurrent Layer. Consider a sequence of inputs $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_T\}$, where T denotes the number of time steps. An RNN computes a corresponding sequence of hidden states $\{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_T\}$ that serve as cumulative summaries of the sequence. At each time step t, the hidden state is updated using a recurrent function f that combines the current input $\vec{x}_t$ with the previous hidden state $\vec{h}_{t-1}$:

$$\vec{h}_t = f(\vec{x}_t, \vec{h}_{t-1}), \tag{2.17}$$

with the initial state $\vec{h}_0$ typically set to the zero vector. This weight sharing across time steps encodes a form of temporal translation invariance, ensuring that the same pattern is processed similarly regardless of its position in the sequence.

The hidden states computed by an RNN can be leveraged in various ways. For instance, if a prediction is required at each time step, a shared MLP can be applied to each $\vec{h}_t$. Alternatively, for tasks such as sequence classification, the final hidden state $\vec{h}_T$ is often used as a fixed-length representation of the entire sequence. In sequence-to-sequence applications like machine translation [22], the final hidden state of an encoder RNN may serve as the initial input to a decoder RNN that generates the output sequence. Additionally, stacking multiple RNN layers, where the hidden states of one layer serve as inputs to the next, forms a "deep" RNN, further enhancing the model's ability to capture complex temporal dependencies. It is worth noting that even a single RNN layer, when unrolled over time, has an effective depth equal to the length of the sequence, which introduces unique training challenges that must be addressed.

Vanishing Gradients. The design of the recurrent cell function f in RNNs is quite flexible, provided it takes as input the current feature vector and the previous hidden state, and outputs an updated hidden state. In the simplest case, this function is implemented as a single-layer MLP, as in the original SimpleRNN model [23, 24], which computes:

$$\vec{h}_t = \sigma(W\vec{x}_t + U\vec{h}_{t-1} + \vec{b}), \tag{2.18}$$

where W and U are learnable weight matrices, $\vec{b}$ is a bias vector, and σ is a nonlinear activation function.

Despite the simplicity of this formulation, SimpleRNNs suffer from significant challenges when applied to long sequences, most notably the vanishing gradient problem. During back-propagation through time, the gradient with respect to an input at an early time step is computed as a product of gradients from subsequent steps. When σ is a sigmoidal function, its derivative is confined to the interval $(0, 1)$, and the repeated multiplication of these values causes the gradients to decay exponentially. As a result, the contributions from earlier time steps diminish to near zero, impeding the network's ability to learn long-range dependencies. For instance, in language tasks where predicting a later word may depend crucially on context provided at the beginning of a sentence, the gradients corresponding to those early inputs become too small to effect meaningful updates.

While the introduction of the ReLU activation function in deep MLPs helped to alleviate the vanishing gradient issue, by maintaining gradients at 0 or 1, the direct application of ReLUs to RNNs is problematic. The unbounded nature of ReLU outputs can lead to exploding gradients, as each time step contributes an unscaled increment, destabilizing training. Thus, the vanishing and exploding gradient problems remain central challenges in the training of recurrent neural networks.

2.1.2.2 Long short-term memory networks

One of the most significant advances in recurrent neural architectures is the long short-term memory (LSTM) network, which was specifically designed to overcome the vanishing gradient problem that plagues simple RNNs. Although other architectures such as the gated recurrent unit (GRU) have been proposed, the LSTM remains one of the most widely used and effective models for capturing long-range dependencies in sequential data.

The key innovation of the LSTM is the introduction of a memory cell that maintains an internal state, denoted by $\vec{c}_t$, across time steps. This memory cell enables gradients to flow unchanged over long sequences, thereby mitigating the vanishing gradient issue. The computation of the LSTM cell involves not only the current input $\vec{x}_t$ and the previous hidden state $\vec{h}_{t-1}$, but also the previous cell state $\vec{c}_{t-1}$. Rather than simply overwriting the previous state with new information, the LSTM selectively retains a portion of it, with the degree of retention learned

directly from data.

Concretely, the LSTM cell employs four distinct linear transformations followed by nonlinearities. First, a candidate cell state $\tilde{\vec{c}}_t$ is computed as

$$\tilde{\vec{c}}_t = \tanh(W_c\vec{x}_t + U_c\vec{h}_{t-1} + \vec{b}_c),$$

which captures the new information to be potentially added to the memory cell. Next, an input gate is used to determine the extent to which this candidate state should be incorporated:

$$\vec{i}_t = \text{logistic}(W_i\vec{x}_t + U_i\vec{h}_{t-1} + \vec{b}_i).$$

Similarly, a forget gate computes the proportion of the previous cell state to be preserved:

$$\vec{f}_t = \text{logistic}(W_f\vec{x}_t + U_f\vec{h}_{t-1} + \vec{b}_f).$$

Finally, an output gate controls how much of the updated cell state is exposed as the new hidden state:

$$\vec{o}_t = \text{logistic}(W_o\vec{x}_t + U_o\vec{h}_{t-1} + \vec{b}_o).$$

Once these gate values are determined, the cell state is updated by combining the retained previous state and the new candidate information:

$$\vec{c}_t = \vec{i}_t \odot \tilde{\vec{c}}_t + \vec{f}_t \odot \vec{c}_{t-1},$$

and the hidden state is then computed as

$$\vec{h}_t = \vec{o}_t \odot \tanh(\vec{c}_t).$$

Since the forget gate is learned from $\vec{x}_t$ and $\vec{h}_{t-1}$, the LSTM dynamically adjusts how much past information to retain, thereby effectively learning to forget irrelevant details over time.

This mechanism enables LSTMs to capture long-range dependencies and maintain robust performance on tasks involving complex sequential data, making them a cornerstone of modern sequence modeling.

Regularising LSTM Networks. Although LSTMs have largely overcome the limitations of standard RNNs in capturing long-range dependencies, their performance remains highly sensitive to the careful tuning and regularisation of their parameters.

Weight initialisation. For initialisation, it is crucial to set the parameters in a manner that facilitates stable training. We can initialise the recurrent weight matrices (denoted by U_*) as orthonormal matrices [25], a choice that preserves the norm of the hidden states as these matrices

are repeatedly applied. In addition, the bias vector for the forget gate, $\vec{b}_f$, is initialised to ones [26] to encourage the network to retain information in its memory cell during the early phases of training, an approach that has been shown to be essential for capturing long-range dependencies. All other weight parameters are initialised using the Xavier initialisation scheme [16], which is particularly well-suited for sigmoidal activations.

Optimisation. Regarding optimisation, adaptive stochastic gradient descent methods such as Adam [19] are typically effective for training recurrent networks. However, due to the repeated application of gradient updates on the same recurrent cell, less aggressive optimisers like RMSprop are sometimes preferred, especially in settings where RNNs are combined with reinforcement learning algorithms.

Regularisation. Regularisation techniques such as dropout and batch normalisation are also applicable to LSTMs, though they require special handling. For example, dropout is most straightforwardly applied to the input-to-hidden transitions [27], while dropout on recurrent connections necessitates using a fixed dropout mask across all time steps [28]. Similarly, while batch normalisation can help stabilize training [29], its direct application to recurrent layers is problematic, leading to the development of alternative normalisation methods such as layer normalisation [30] and weight normalisation [31]. Despite these advances, the design of effective normalisation schemes for recurrent networks continues to be an active area of research.

2.1.3 Neural networks for graphs

A significant portion of this thesis addresses models that process data combining sequential and graph-structured information, a hallmark of multimodal data. Graph data inherently generalise many common input types such as images, text, and speech, making the design of suitable neural layers for graphs one of the prominent challenges in contemporary machine learning [32–34].

To illustrate this, I focus on Graph Neural Networks (GNNs), particularly in the context of node classification. In this setting, the input is typically represented by a matrix of node features, $X \in \mathbb{R}^{N \times F}$, where each of the N nodes is associated with F features, along with an adjacency matrix $A \in \mathbb{R}^{N \times N}$ that encodes the connectivity between nodes. The objective is to assign each node to one of C classes, thereby producing a matrix of class probabilities, $Y \in \mathbb{R}^{N \times C}$, where the entry Y_{ij} represents the probability that node i belongs to class j.

By leveraging the intrinsic structure of graphs, GNNs enable the propagation and aggregation of information across nodes, facilitating the learning of representations that capture both local and global relationships. This capability is central to the effective classification of nodes, as it allows the model to integrate feature information with the topology of the graph, thereby enhancing predictive performance on tasks where the relational structure is crucial.

2.1.3.1 Graph neural networks

Graph Neural Networks (GNNs) are a class of neural models designed to operate on graph-structured data. Initially proposed by [35, 36], GNNs leverage the inherent structure of graphs, incorporating node features X_v and edge features $X_{(u,v)}$, to learn rich representation vectors for individual nodes, $\vec{h}_v$, and for entire graphs, $\vec{h}_G$. These networks typically follow a recursive message passing or neighborhood aggregation framework [37, 38], in which the representation of a node v at iteration k, denoted by $\vec{h}_v^k$, is updated by aggregating the representations of its neighbors from the previous iteration. The initial representations $\vec{h}_v^0$ are set to the input node features, i.e., $\vec{h}_v^0 = \vec{X}_v$. After k iterations of message passing, each node's representation encapsulates information from its k-hop neighborhood.

This process can be formally described by two operations. First, an aggregation function gathers information from the neighbors of a node:

$$\vec{a}_v^k = \text{AGGREGATE}^k \left(\{ \vec{h}_u^{k-1} : u \in \mathcal{N}(v) \} \right),$$

and then a combine function integrates this aggregated information with the node's previous representation:

$$\vec{h}_v^k = \text{COMBINE}^k \left(\vec{h}_v^{k-1}, \vec{a}_v^k \right).$$

When edge features are available, a similar formulation is employed by incorporating the edge information along with the representations of the connected nodes:

$$\vec{h}_v^k = \text{AGGREGATE}^k \left(\{ (\vec{h}_u^{k-1}, \vec{h}_v^{k-1}, \vec{X}(u,v)) : u \in \mathcal{N}(v) \} \right).$$

For tasks such as node classification or edge prediction, the final node representations $\vec{h}_v^K$ (obtained after K iterations) are used for prediction. In the case of graph-level prediction, a READOUT function is applied to the set of node representations to yield a single vector for the entire graph:

$$\vec{h}_G = \text{READOUT} \left(\{ \vec{h}_v^K \mid v \in G \} \right).$$

The READOUT function is typically chosen to be a permutation invariant operation, such as summation, although more sophisticated functions can also be employed. In the following sections, several GNN architectures relevant to the methods proposed in this thesis will be explained in greater detail.

2.1.3.2 Graph convolutional networks

Graph Convolutional Networks (GCNs) were among the first architectures to directly incorporate graph structure into neural network design. In a GCN, the aggregation of information from a node's neighborhood is typically performed using an element-wise mean pooling operation. Let

W^k denote the learnable weight matrix at the k-th layer and define the ReLU activation function as $\text{ReLU}(x) = \max(0, x)$. In the simplest formulation, the update rule for the representation of a node v is given by:

$$\vec{h}_v^k = \text{ReLU}\left(W^k \cdot \text{MEAN}\left\{\vec{h}_u^{k-1} : u \in \mathcal{N}(v) \cup \{v\}\right\}\right).$$

Here, the set $\mathcal{N}(v) \cup \{v\}$ denotes the union of the neighbors of v and v itself, ensuring that a node's own features are incorporated in the aggregation.

An alternative update rule employs a normalisation scheme that accounts for the degrees of the nodes. Defining $\tilde{\mathcal{N}}(v) = \mathcal{N}(v) \cup \{v\}$ and letting $\deg(v)$ denote the degree of node v, the update becomes:

$$\vec{h}_v^k = \text{ReLU}\left(W^k \cdot \sum_{u \in \tilde{\mathcal{N}}(v)} (\deg(v)\deg(u))^{-1/2} \vec{h}_u^{k-1}\right).$$

This normalisation ensures that the contributions from nodes are appropriately scaled by their degrees, which helps to stabilize the learning process in graphs with nodes of varying connectivity.

2.1.3.3 Graph attention networks

Graph Attention Networks (GATs) incorporate the attention mechanism [39] into the propagation framework for graphs, allowing the network to learn the importance of neighboring nodes when updating a node's representation. In a GAT, an initial shared linear transformation is applied to the input features of every node, using a weight matrix $W \in \mathbb{R}^{F' \times F}$, where F is the dimension of the input features and F' is the dimension of the transformed features. This is followed by a self-attention mechanism that computes attention coefficients between nodes.

Specifically, for each pair of nodes i and j, the attention mechanism defines a function $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$ such that the non-normalised attention coefficient is given by

$$e_{ij} = a\left(W\vec{h}_i, W\vec{h}_j\right).$$

In practice, the function a is often implemented as a single-layer feed-forward neural network parameterized by a weight vector $\vec{a} \in \mathbb{R}^{2F'}$, followed by a LeakyReLU nonlinearity. The graph structure is incorporated by computing e_{ij} only for nodes j in the neighborhood $\mathcal{N}_i$ of node i. These coefficients are then normalised across the neighborhood using the softmax function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}.$$

A common formulation for the attention coefficients is therefore:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T \left[W\vec{h}_i \parallel W\vec{h}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{a}^T \left[W\vec{h}_i \parallel W\vec{h}_k\right]\right)\right)},$$

where $\parallel$ denotes the concatenation of vectors.

Once the normalised attention coefficients are computed, they are used to aggregate the features from the neighborhood, yielding the updated node representation:

$$\vec{h'}_i = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij} W\vec{h}_j\right),$$

with σ representing a non-linear activation function.

To improve the stability and expressive power of the model, multi-head attention is often employed. In this setting, K independent attention mechanisms compute their own set of attention coefficients and transformed features, which are then concatenated to form the final node representation:

$$\vec{h'}_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k W^k\vec{h}_j\right),$$

where $\parallel$ denotes concatenation, α_{ij}^k are the normalised attention coefficients computed by the k-th head, and W^k is the corresponding weight matrix.

In cases where the final layer of the network is used for prediction, concatenation is less appropriate due to the increased dimensionality. Instead, the outputs from each head are averaged:

$$\vec{h'}_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k W^k\vec{h}_j\right).$$

This formulation ensures that the final output retains a consistent dimension while still capturing the benefits of multi-head attention.

2.2 Modern architectures for language modelling

2.2.1 The Transformer

The evolution of neural network architectures for sequence processing reached a critical turning point with the introduction of the Transformer [1], which fundamentally addressed the long-standing computational limitations of recurrent neural networks (RNNs) and long short-term memory (LSTM) networks. Prior to the Transformer, sequential models struggled with processing long-range dependencies and faced significant computational constraints that hindered their

ability to capture global contextual information.

The breakthrough of the Transformer architecture lies in its revolutionary self-attention mechanism, which fundamentally reimagines how neural networks process sequential data [40]. Unlike recurrent models that process inputs sequentially, the self-attention mechanism enables every element in a sequence to interact directly with all other elements, irrespective of their positional distance. This capability allows for a more sophisticated and efficient representation of contextual relationships within the data.

At the heart of the Transformer's innovative design are several key architectural components. In the sections that follow, I detail the self-attention mechanism, positional encodings, and the various Transformer configurations, namely the encoder-only, decoder-only, and encoder-decoder variants.

2.2.1.1 Self-attention mechanism

The self-attention mechanism is the core innovation of the Transformer model. It replaces the inherently sequential operations of RNNs with a fully parallelizable approach, allowing the model to compute interactions among all tokens in the input sequence simultaneously. This mechanism dynamically weighs the relevance of each token with respect to every other token.

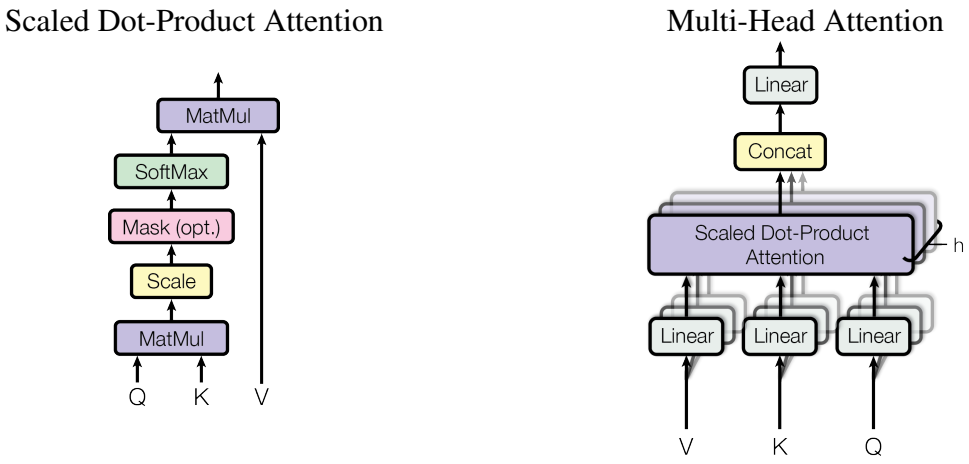

Figure 2.1: (left) Scaled Dot-Product Attention. **(right)** Multi-Head Attention consists of several attention layers running in parallel. Figure extracted from [1].

Mathematically, the self-attention mechanism is formulated as follows. Given an input sequence, three matrices are derived via learned linear projections: the Query matrix Q, the Key matrix K, and the Value matrix V. The output of the attention layer is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

where d_k denotes the dimensionality of the key (and query) vectors. The scaling factor $\sqrt{d_k}$ mitigates the risk of exceedingly large dot product values that could lead to vanishing gradients when applying the softmax function.

Furthermore, the concept of multi-head attention extends the self-attention mechanism by executing multiple attention operations (or "heads") in parallel. Each head processes different linear projections of the input, allowing the model to attend to diverse aspects of the data simultaneously. The outputs of these heads are concatenated and linearly transformed to produce the final output of the layer, thereby enriching the representational capacity of the model.

2.2.1.2 Positional encodings

Since the self-attention mechanism does not inherently encode the sequential order of the tokens, the Transformer incorporates positional encodings to imbue the model with information regarding the order of tokens. These encodings are added to the input embeddings and are crucial for preserving the spatial and sequential context.

The original Transformer employs sinusoidal positional encodings defined as:

$$\text{PE}(pos, 2i) = \sin\Big(\frac{pos}{10000^{2i/d_{model}}}\Big),$$

$$\text{PE}(pos, 2i+1) = \cos\Big(\frac{pos}{10000^{2i/d_{model}}}\Big),$$

where pos represents the token position in the sequence, i denotes the dimension index, and d_{model} is the hidden dimensionality of the model. This formulation produces encodings that vary smoothly across positions, ensuring that the model can capture relative positional relationships even for sequences longer than those encountered during training.

2.2.1.3 Encoder and decoder

The Transformer architecture is typically composed of two principal modules: the encoder and the decoder. Figure 2.2 provides an overview of the encoder and decoder structures. Both modules are constructed as stacks of identical layers, but they serve distinct roles.

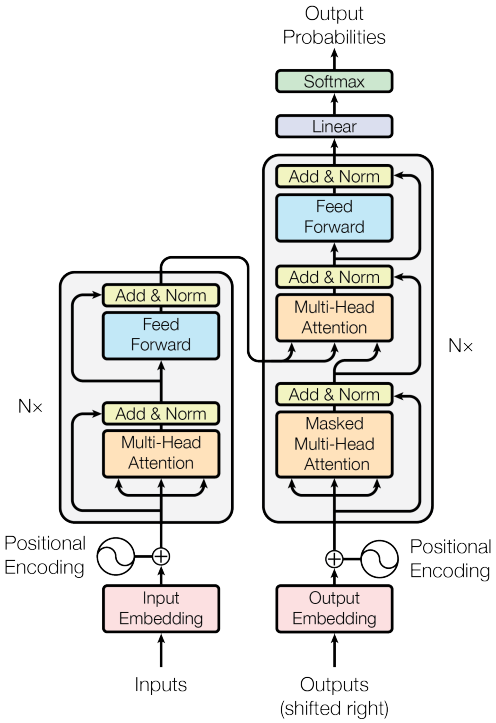

Figure 2.2: Architecture of the Transformer model. Figure extracted from [1].

Encoder. The encoder processes the input sequence by applying a series of identical layers, each of which comprises two main sub-modules: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. Residual connections and layer normalisation are applied around each sub-module to facilitate training and improve gradient flow. The encoder is responsible for extracting comprehensive features from the input data, enabling the model to capture both local and global dependencies.

Decoder. The decoder generates the output sequence based on the encoded representation of the input. Each decoder layer consists of three sub-modules: (1) a masked multi-head self-attention mechanism that ensures the model cannot access future tokens during training, (2) a multi-head attention module that attends to the encoder's output, and (3) a position-wise feed-forward network. The inclusion of a masking mechanism in the decoder's self-attention layer is critical to prevent the model from "cheating" by using information from subsequent tokens during sequence generation.

2.2.1.4 Encoder-only Transformer

The encoder-only variant of the Transformer architecture comprises solely encoder layers and is typically employed in tasks that require understanding and representation of the input sequence without generating a corresponding output sequence. This configuration is particularly effective for applications such as text classification, sentence embedding, and other discriminative tasks.

A prominent example of an encoder-only Transformer is BERT (Bidirectional Encoder Representations from Transformers) [41], which is pre-trained on large corpora using unsupervised objectives (e.g., masked language modeling and next sentence prediction) and later fine-tuned for various downstream tasks. The encoder-only design has also been successfully applied beyond natural language processing, such as in computer vision tasks where the model is used as a feature extractor.

2.2.1.5 Decoder-only Transformer

In contrast, the decoder-only Transformer architecture consists exclusively of decoder layers. This configuration is predominantly used in autoregressive tasks, where the objective is to generate a coherent output sequence conditioned on a given context. Decoder-only models are well-suited for language modeling and text generation, as each token is predicted based on the preceding tokens.

Recent large-scale language models (LLMs) such as GPT [42], GPT-2, GPT-3 [3], and GPT-4 [43], as well as models like LLaMA [44] and Mistral [45], exemplify the decoder-only architecture. Their autoregressive training objectives enable the generation of contextually relevant and coherent text, facilitating applications in dialogue systems, content creation, and beyond.

2.2.1.6 Encoder-decoder Transformer

The encoder-decoder Transformer, originally proposed by Vaswani et al. [1], integrates both encoder and decoder modules into a unified architecture. This design is specifically tailored for sequence-to-sequence tasks, such as machine translation, text summarization, and conversational modeling.

In this architecture, the encoder first processes the entire input sequence to generate a rich contextual representation. The decoder then leverages this representation, along with its own masked self-attention mechanism, to produce the output sequence. Notable models that employ the encoder-decoder framework include T5 (Transfer Text-to-Text Transformer) [46] and BART [47], which have demonstrated remarkable performance on a variety of generative tasks.

The encoder-decoder paradigm remains a powerful and flexible approach for tasks that require a nuanced transformation from one sequence to another, making it one of the most influential designs in modern deep learning research.

2.2.2 Large language models

Large Language Models (LLMs) represent a sophisticated computational approach to understanding and generating human language. By learning probabilistic distributions over extensive textual corpora, these models capture intricate linguistic nuances that underpin modern natural

language processing (NLP) applications [2]. The advent of Transformer-based architectures has catalyzed a paradigm shift in language modeling, replacing traditional statistical methods with advanced deep learning techniques that dynamically encode contextual information.

2.2.2.1 From static to contextual representations

Historically, language models relied on static, context-independent word embeddings that assigned a single representation to each word. In contrast, modern Transformer-based models, exemplified by BERT, GPT, and their numerous variants, employ self-attention mechanisms to generate dynamic, contextually aware representations [48]. This innovation allows each token's embedding to adapt according to its surrounding linguistic environment, thereby capturing the polysemous nature of language. Such contextual embeddings better reflect the complexities of human language comprehension, where the meaning of a word is often influenced by the entire sentence or discourse.

2.2.2.2 Pre-training and fine-tuning paradigm

A hallmark of contemporary LLMs is their two-stage training process. The initial *pre-training* phase leverages massive, diverse text collections to enable models to learn broad linguistic patterns through unsupervised learning objectives. Typical pre-training strategies include masked language modeling, where certain tokens are masked and then predicted, and next sentence prediction, which fosters an understanding of broader contextual relationships [3].

Following pre-training, models undergo a *fine-tuning* phase, wherein they are adapted to specific downstream tasks. This stage involves supervised learning on relatively limited task-specific datasets, allowing the pre-trained models to achieve remarkable performance across diverse applications such as text classification, named entity recognition, question answering, and machine translation.

2.2.2.3 Architectural innovations and their impact

The Transformer-based architecture has profoundly impacted the design and efficacy of LLMs. The self-attention mechanism, a core component of the Transformer, enables efficient processing of long-range dependencies by allowing every token to attend to every other token within a sequence. This mechanism facilitates the capture of complex semantic and syntactic relationships, a capability that is further enhanced by the use of multi-head attention.

The impact of these innovations extends well beyond theoretical advancements. In practice, Transformer-based language models have set new performance benchmarks in a wide array of NLP tasks [46, 49]. Their ability to generate coherent and contextually appropriate text has led to transformative applications in fields ranging from conversational AI to content generation and beyond.

2.3 Multimodal learning in large language models

Multimodal learning represents a critical paradigm in artificial intelligence that aims to integrate heterogeneous data sources into a unified framework. This approach mirrors the complex sensory integration inherent to human cognition, enabling the simultaneous processing of diverse modalities such as text, vision, audio, and graph-structured data. By combining these distinct forms of information, multimodal systems capture richer, complementary representations that facilitate improved reasoning and understanding compared to single-modal models [50, 51].

2.3.1 Beyond single-modal learning

Traditional machine learning methods have largely focused on processing single modalities. For instance, natural language processing models exclusively handle text, while computer vision systems are designed solely for image inputs. Although these single-modal approaches have achieved significant success within their respective domains, they inherently limit a system's ability to capture the full spectrum of real-world information. Human perception, by contrast, relies on the simultaneous integration of visual cues, auditory signals, and linguistic context to form a comprehensive understanding of the environment. Inspired by this, multimodal learning seeks to replicate such integration in artificial systems, thereby enabling the development of large language models that can reason over multiple data types concurrently and achieve a more nuanced representation of complex scenarios [52, 53].

2.3.2 Architectural foundations of multimodal language models

Recent advances in Transformer-based architectures have provided a robust foundation for developing multimodal large language models. These models extend the core principles of self-attention to accommodate multiple data modalities. The following subsections elaborate on the key components of these architectures.

2.3.2.1 Modality-specific encoders

In multimodal LLMs, each data modality is processed by a dedicated encoder that transforms raw inputs into latent representations. Textual information is typically encoded using models such as BERT or GPT [2], which generate contextual embeddings that capture semantic nuances. Visual inputs are handled by vision encoders, such as Vision Transformers or convolutional neural networks [54], which extract spatial and structural features from images. Similarly, audio data may be processed using recurrent networks or temporal convolutional networks, and graph-structured data is often encoded using graph neural networks. These modality-specific encoders ensure that the unique characteristics of each data type are effectively captured before integration.

2.3.2.2 Cross-modal fusion mechanisms

After obtaining modality-specific representations, the next challenge is to align and integrate these diverse data sources into a unified latent space. Cross-modal fusion mechanisms extend the self-attention paradigm of Transformers to allow the model to attend across modalities. This approach enables the system to learn the intricate inter-modal dependencies that are essential for capturing a comprehensive view of the data. For instance, cross-modal attention mechanisms have been successfully employed in models such as Flamingo [55], where visual and textual inputs are jointly processed to facilitate coherent reasoning. By effectively fusing information from multiple sources, these mechanisms contribute significantly to the improved performance of multimodal LLMs.

2.3.2.3 Unified multimodal Transformer architectures

The ultimate goal of integrating modality-specific encoders and cross-modal fusion techniques is to develop a unified Transformer architecture capable of processing multimodal inputs within a single framework. Such unified architectures overcome the limitations inherent in single-modal models by providing a holistic representation that encapsulates information from text, images, audio, and even structured data. Recent models, such as PaLM-E [56], demonstrate the feasibility of this approach by leveraging cross-modal attention and joint representation learning to achieve superior performance on complex tasks. These architectures represent a significant step towards models that more closely mimic human-like understanding, by effectively bridging the gap between diverse data sources.

2.4 Machine learning tasks

This thesis investigates three fundamental machine learning tasks that advance the state-of-the-art in multimodal learning and knowledge representations. Each task presents unique challenges in processing and leveraging different types of data modalities while maintaining practical applicability in real-world scenarios.

2.4.1 Text-attributed node classification on hypergraphs

Text-attributed node classification on hypergraphs represents a significant advancement over traditional graph learning techniques by integrating rich textual information with complex hypergraph structures. In contrast to conventional methods that primarily exploit the graph topology in terms of binary relations, this approach leverages both the higher-order structural relationships (as n-ary edges) among nodes and the semantic content associated with them, thereby enabling more nuanced and effective learning where high order relations between nodes exists.

2.4.1.1 Problem formulation

Formally, let us consider a text-attributed hypergraph $\mathcal{H} = (V, E, X)$, where $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of nodes, $E = \{e_1, e_2, \ldots, e_m\}$ is the set of hyperedges with each $e_i \subseteq V$, and $X = \{x_1, x_2, \ldots, x_n\}$ comprises the textual content associated with each node v_i, with $x_i \in \mathcal{X}$. The goal of node classification is to learn a function

$$f : \mathcal{X} \times \mathcal{G} \to \mathcal{Y},$$

which maps each node, taking into account both its textual features and the hypergraph structure, to a corresponding label $y \in \mathcal{Y}$. Here, $\mathcal{G}$ represents the space of all possible hypergraph structures.

A central challenge in this task is to capture and integrate two distinct forms of information. On one hand, the hypergraph's incidence matrix $H \in \{0, 1\}^{|V| \times |E|}$ encodes the explicit structural connections among nodes, reflecting higher-order interactions that extend beyond simple pairwise relationships. On the other hand, the semantic content of each node is represented by high-dimensional embeddings in $\mathbb{R}^d$, which capture the nuanced meaning contained in the text. The task thus requires developing architectures that can effectively process and fuse these complementary sources of information.

2.4.1.2 Approach overview

To address these challenges, my approach leverages recent advances in language modeling while extending them to handle hypergraph-structured data. Specifically, I propose a framework that jointly learns representations by combining a text encoder with a hypergraph message-passing operator. This can be formally expressed as

$$h_i = \phi\Big(\psi(x_i),\ \mathrm{AGG}(\{h_j : v_j \in \mathcal{N}(v_i)\})\Big),$$

where ψ is a text encoder that transforms the raw textual input into a semantic embedding, AGG denotes a hypergraph aggregation function that captures structural relationships from the neighborhood $\mathcal{N}(v_i)$ of node v_i, and ϕ is a learnable function that fuses the semantic and structural representations. This joint representation is then used for robust node classification, demonstrating improved performance in real-world applications where capturing higher-order interactions is critical.

2.4.2 Semantic matching for dense retrieval

Semantic matching represents a fundamental challenge in information retrieval systems, encompassing both the identification of semantic relationships and the verification of factual claims [57]. While traditional information retrieval approaches have focused on lexical matching [58],

modern systems must handle increasingly complex scenarios involving different data modalities and sophisticated reasoning requirements [59]. This section presents a formal framework for semantic matching in the context of dense retrieval, with particular emphasis on cross-modal scenarios involving structured knowledge bases and natural language queries.

2.4.2.1 Problem formulation

Semantic matching in a cross-modal setting aims to bridge the gap between different representational formats while preserving semantic meaning [60]. Given a query space $\mathcal{Q}$ and a knowledge base $\mathcal{K}$, I seek to learn a function that maps query-evidence pairs to relevance scores. The complexity of this task stems from the inherent differences in how information is encoded across modalities - while queries follow sequential linguistic patterns, knowledge bases often organize information through structured representations [61].

In my framework, I specifically consider the alignment between natural language queries and knowledge graph representations, building on recent advances in graph neural networks [62] and cross-modal learning [63]. Formally, the knowledge base is structured as a graph $G = (V, E, R)$ where V is the set of entities, $E \subseteq V \times V$ is the set of edges, and R is the set of relation types. The queries are represented as a set $Q = \{q_1, ..., q_n\}$ where each $q_i \in \mathcal{Q}$ represents a claim or question. The goal is to learn a matching function $m : \mathcal{Q} \times \mathcal{K} \rightarrow [0, 1]$ that assigns relevance scores to query-evidence pairs.

The alignment process, drawing from advances in representation learning [64], must address challenges at multiple levels of granularity. At the local level, individual concepts and entities must be matched across modalities - for instance, recognizing that a textual description refers to a specific entity in the knowledge graph [65]. At the global level, the system needs to understand how these concepts relate to each other, preserving the structural information encoded in the knowledge graph while making it comparable with the natural language expression of relationships in queries [66].

This cross-modal alignment presents several key challenges that have been identified in recent literature. Firstly, semantic understanding requires the system to capture the nuanced meaning embedded within natural language queries while preserving structured relationships from the knowledge base [41]. Secondly, structural reasoning introduces complexity, as the matching process must accommodate not only direct matches but also indirect relationships necessitating multi-hop reasoning across interconnected knowledge graph nodes [67]. Finally, effective representation learning is crucial; the system must develop compatible embeddings for textual and graph-based data to facilitate meaningful comparisons, overcoming inherent representational disparities between modalities [68].

2.4.2.2 Application to claim verification

Semantic matching principles can be directly applied to the task of claim verification, where the goal is to determine whether a given claim is supported by evidence from a reliable knowledge source [69]. A prominent example is the Fact Extraction and VERification (FEVER) task, which requires systems to both retrieve relevant evidence and determine whether it supports or refutes a given claim [69].

In the FEVER setting, claims are presented as natural language statements, while evidence must be retrieved from Wikipedia articles. This creates a challenging cross-modal scenario where systems must bridge the gap between unstructured claims and semi-structured article content [70]. The task is further complicated by the need to sometimes combine multiple pieces of evidence to verify a single claim, requiring sophisticated reasoning capabilities [71].

Beyond simple relevance matching, claim verification systems must determine the nature of the relationship between claim and evidence: whether the evidence supports, refutes, or is insufficient to verify the claim [70]. This makes claim verification a more complex application of semantic matching that requires additional reasoning capabilities and careful consideration of logical relationships between query and evidence [72, 73].

2.4.3 Tabular data augmentation with pretrained Transformers

Tabular data represents a distinct and often overlooked modality in machine learning, despite its prevalence in critical domains such as healthcare, physics, and finance. While data augmentation has proven highly effective for improving model performance in computer vision and natural language processing, its application to tabular data presents unique challenges. These challenges stem primarily from the lack of clear symmetries in the input space and the complex interdependencies between features that characterize tabular data structures.

2.4.3.1 Problem formulation

Given a tabular dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where $\mathbf{x}_i \in \mathbb{R}^d$ represents a sample with d features and $y_i \in \mathcal{Y}$ its corresponding label, the task of tabular data augmentation aims to generate synthetic samples that preserve the underlying data distribution while introducing meaningful variations. Traditional augmentation techniques that rely on simple transformations or perturbations often fail in the tabular domain, as they can disrupt critical feature relationships and lead to unrealistic or invalid samples.

The fundamental challenge lies in identifying invariant transformations that respect both the local structure of individual features and the global relationships between them. Unlike images or text, where transformations like rotations or synonym substitutions preserve semantic meaning, tabular data lacks such obvious symmetries. Moreover, features in tabular data often

exhibit mixed types (continuous, categorical, ordinal) and complex dependencies that must be maintained during the augmentation process.

2.4.3.2 Transformer-based manifold data augmentation

My work addresses these challenges through a novel approach that leverages Transformer architectures and manifold learning principles. The key insight is that by mapping tabular data into a learned manifold space using pre-trained Transformers, I can perform augmentation operations that inherently respect the data's structural properties. Specifically, I utilize a pre-trained in-context model to encode tabular samples into a high-dimensional embedding space:

$$\phi : \mathbb{R}^d \to \mathbb{R}^{d'}, \quad \mathbf{z} = \phi(\mathbf{x}) \tag{2.19}$$

where ϕ represents the Transformer encoder and $\mathbf{z}$ is the resulting embedding in the manifold space.

The innovation in my approach lies in its training-free nature and the use of in-context learning principles. Rather than requiring expensive model training or complex generative architectures, my method leverages the rich representational capabilities of pre-trained Transformers. This allows to capture both local and global patterns in the data while maintaining computational efficiency. The augmentation process involves three steps, namely: 1) mapping samples to the manifold space using the Transformer encoder, 2) performing controlled perturbations in this space while preserving label-invariant properties, and 3) generating diverse yet realistic synthetic samples through manifold-guided transformations.

2.4.3.3 Application to downstream tasks

The practical significance of my approach extends beyond theoretical novelty. By enabling effective augmentation of tabular data, I address a critical need in many real-world applications where data scarcity limits model performance. My method demonstrates significant improvements in classification tasks across various domains, particularly in scenarios with limited training data.

Particularly, the training-free nature of my approach makes it valuable in modern AI systems that require scalable and efficient solutions. Unlike traditional augmentation techniques that may require extensive hyperparameter tuning or computational resources, my Transformer-based method provides a practical pathway for improving model performance while maintaining efficiency. This represents an important step toward making advanced data augmentation techniques accessible and practical for tabular data applications.

CHAPTER 3

HIGHER-ORDER REPRESENTATIONS FOR TEXT-ATTRIBUTED NODE CLASSIFICATION

3.1 Introduction and contribution overview

In recent years, the study of hypergraphs has gained significant attention in the field of network analysis [74–76]. Unlike traditional graphs, where edges represent pairwise relationships, hypergraphs are a generalisation of graphs where an edge can join any number of nodes, allowing for more complex interactions by connecting multiple nodes through hyperedges. This characteristic makes hypergraphs an ideal framework for representing complex systems such as social networks, biological networks, and collaboration networks, where relationships often involve sets of entities.

Machine learning on hypergraphs [74, 77], has been shown to be an effective tool for studying complex graph-based data structures. In particular, when compared with a standard graph, a hypergraph is more capable and flexible in modeling high-order relations since one hyperedge in a hypergraph can connect more than two nodes. As a result, hypergraph learning has recently gained increasing attention and has been applied to many research fields, such as computer vision [78, 79] and recommendation systems [80, 81].

In many real-world complex systems, hypergraph data is accompanied by node attribute information [82–85]. Some examples of node attributes include the age, ethnicity, and gender of individuals in social networks [86], affiliations of authors in collaboration networks [87], and article abstract texts in co-citation graphs [88]. Previous studies have demonstrated that node attribute data potentially enhances the learning of community structure in networks [89, 90].

A particular instance of attributed hypergraphs are Text-Attributed Hypergraphs (TAHGs) [82–85], which have been widely used for modeling a variety of real-world applications, such as co-authoring in collaboration networks [91, 92], information retrieval [85, 93], product recommendation [94], and many others.

Given the prevalence of TAHGs [82–85], I have explored how to effectively handle these graphs, with a focus on node classification tasks. Such tasks aims at predicting the category or label of a node based on its textual attributes and the structure of the hypergraph. This problem is pivotal in various applications such as predicting protein functions in biological networks [95], identifying roles in social networks [91, 92], and classifying documents in citation networks [88]. Inherently, TAHGs provide both node attribute and graph structural information. Thus, it is critical to effectively capture both while modeling their interrelated correlation for effective representation learning.

Node classification in text-attributed hypergraphs presents unique challenges that are not encountered in traditional graph-based approaches [83, 96, 97]. In this context, there are two key hurdles. First, since hyperedges can connect any set of nodes, capturing such complex interaction patterns becomes increasingly difficult using conventional graph-based methods. Second, effectively exploiting the textual information to learn informative node representations is complex. Recent advances in the field of NLP have led to the emergence of language models (LMs) with strong capabilities for semantic understanding of text [42, 98–101]. However, the integration of hypergraph structural information directly into language models is still an open and under-researched topic.

To address the above limitations, I propose a novel architecture, HyperBERT, which mixes hypergraph-aware layers into language models to simultaneously exploit hypergraph topology and textual representations for the node classification task on text-attributed hypergraphs. In particular, HyperBERT extends the original BERT architecture with hypergraph-specific inductive biases. To do so, I design specially-crafted hypergraph-aware layers into the BERT encoder, effectively mixing hypergraph structure with text semantic embeddings for computing node representations. To accomplish the text-hypergraph alignment in the feature space, I propose a novel self-supervised loss for pretraining HyperBERT and effectively aligning semantic and hypergraph feature spaces. HyperBERT achieves state-of-the-art performance across five widely used hypergraph benchmarks. My code is available at https://github.com/AdrianBZG/HyperBERT.

3.2 Related work

My work is related to two lines of research: text-attributed graph learning-based methods and language model-based approaches on text-attributed graphs [85, 102, 103]. In this section, I review briefly the relevant literature for such categories.

Given the recent success of graph representation learning, numerous research works have been proposed for various tasks including node classification [104] or link prediction [105]. Graph Neural Networks (GNNs) are recognized as the de facto standard technique for modeling graph data. These methods (e.g. GCN [62], GAT [106], GraphSAGE [107] or GIN [108]) learn effective mechanisms such that information between the nodes is aggregated for expressive

graph representations. In the case of TAGs, a cascade architecture [107] is typically adopted. In such setting node features are encoded independently using text modeling techniques (e.g. Bag-of-Words [109], skip-gram [110], or more recently language models such as BERT [98]), and subsequently aggregated via the message passing mechanism to produce the final node representations. However, standard GNN-based approaches are only applicable to general graphs with binary relations.

To alleviate this issue recent research has focused on developing effective methodologies for attributed hypergraph learning in order to capture n-ary or high-order interactions between nodes [80, 111–113]. For instance, [114] proposes an inductive multi-hypergraph learning algorithm, which learns an optimal hypergraph embedding with good performance on the task of multi-view 3D object classification. More recently, several hypergraph-based methods have been proposed for the task of node classification [74, 115]. In particular, [116] propose the Hypergraph Neural Network (HGNN) architecture, which encodes high-order node correlations using a hypergraph structure and generalises the convolution operations to the hypergraph learning process based on hypergraph Laplacian and truncated Chebyshev polynomials. [74] propose to supplement the family of graph neural networks with two end-to-end trainable operations, i.e. hypergraph convolution and hypergraph attention (HCHA). [115] propose HyperGCN, model to enhance the hypergraph Laplacian with additional weighted pair-wise edges. [117] propose HypeBoy, a hypergraph self-supervised learning method for node classification. However, HypeBoy primarily focuses on hypergraphs without specifically addressing the complexities introduced by nodes with textual attributes.

Given the success of LMs in solving semantic representation learning tasks, recent works [118] have proposed to utilise LMs for TAGs, as they can transform the nodes' textual attributes into semantically rich node embeddings. However, node embeddings provided by LMs often only contain information from the textual attributes, overlooking the topological structure present in the TAGs. To overcome this limitation, multiple works integrate LM-based embeddings with graph learning methods [119, 120], with the goal of generating node embeddings that are tailored for specific TAG tasks.

For instance, [85] propose Graphormer, which fuses LM-based text encoding and graph aggregation into an iterative workflow. GIANT [121] introduces a novel neighborhood prediction task to fine-tune a XR-Transformer [122], and feeds node embeddings as obtained by the LM into GNNs for downstream tasks. TAPE [123] presents a method that uses LMs to generate prediction text and explanation text, which serve as augmented text data to subsequently feed into GNNs. LM-GNN [124] introduces graph-aware pre-training to adapt the LM on the given graph before fine-tuning the entire model, demonstrating significant performance results. SimTeG [125] finds that first training the LMs on the downstream task and then freezing the LMs and training the GNNs can result in improved performance. K-BERT [126] leverages a knowledge layer to inject relevant triplets from a knowledge graph into the input sentence and transform

it into a knowledge-rich sentence tree. KG-BERT [127] treats triples in knowledge graphs as textual sequences. They propose to model entity and relation descriptions of a triple as input, and compute the scoring function of the triple with the KG-BERT language model. Despite these advances, previous works do not take into account the problem where both text attributes and higher-order interactions exist between the entities, as is the case with text-attributed hypergraph node classification.

3.3 Preliminaries

In this paper, I focus on learning representations for nodes in Text-Attributed Hypergraphs (TAHG), where I take node classification as the downstream task. Therefore, before describing the details of my proposed method, I start with presenting a few basic concepts, including the formal definition of TAHGs and how language models can be used for node classification in TAHGs. Then, I introduce the formulation of the problem my method is tackling.

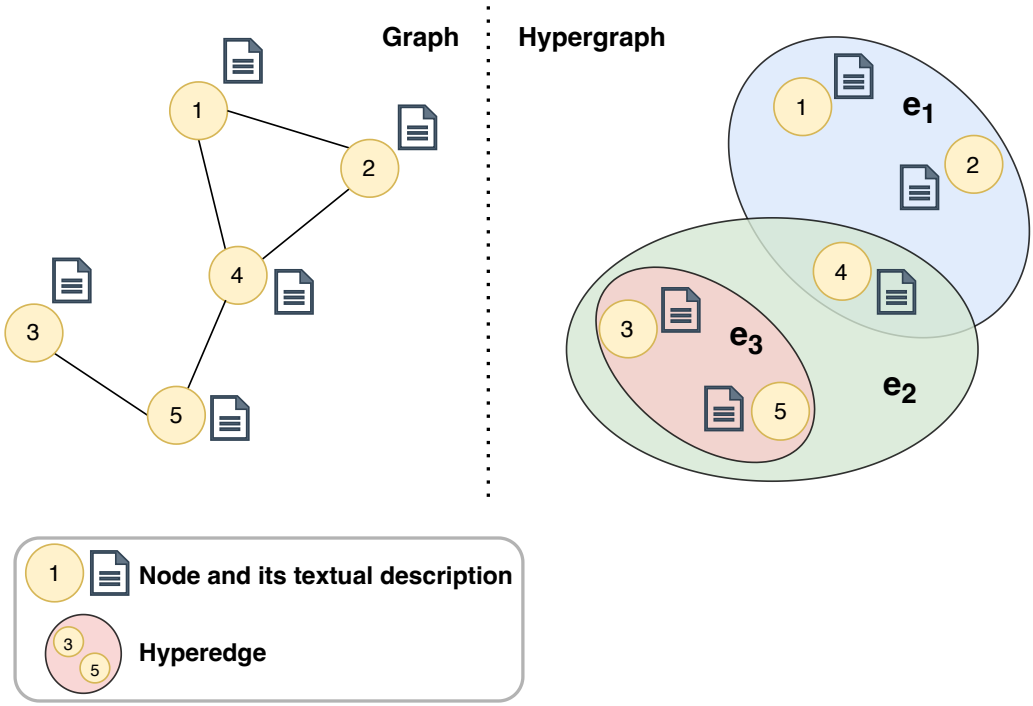

Figure 3.1: An illustration of a standard text-attributed graph **(left)** where nodes are connected between each other using binary relations, and a text-attributed hypergraph **(right)** where nodes are related with high-order connections (hyperedges). In both cases, each node is attributed with a textual description, such as paper abstract in the case for co-citation hypergraphs.

3.3.1 Text-attributed hypergraphs

A hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{S}^V)$ is defined by a node set $\mathcal{V}$ and a hyperedge set $\mathcal{E}$. Each hyperedge $e_j \in \mathcal{E}$ is a non-empty set of nodes. Each node $v_i \in \mathcal{V}$ is associated with a sequential text feature,

$s_i \in \mathcal{S}^V$, and the corresponding label y_i. Each label has a real label text c from the set of all label texts C (e.g. 'Artificial Intelligence' or 'Supervised Learning' for the case of paper categories in a co-citation hypergraph). For the sake of clarity, Figure 3.1 illustrates the difference between standard text-attributed graphs and text-attributed hypergraphs.

3.3.2 Pretrained language models for node classification

In the context of TAGs, LMs can be employed to encode the text attributes associated with each node and learn a representation that captures the semantic meaning of the text. Let $s_i \in \mathcal{S}^V$ denote the text attributes of node i, and LM be a pre-trained network, such as BERT [98]. Then, the text attributes of node i can be encoded by applying BERT to s_i. The output of the LM is denoted as $\mathbf{z_i} \in \mathbb{R}^d$, where d is the dimensionality of the output feature vector. Then, to perform node classification, the output is typically fed into a fully connected layer with a softmax function for class prediction. The goal is to learn a function that maps the encoded text attributes to the corresponding node labels.

3.3.3 Problem formulation

Given a text-attributed hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{S}^V)$, with each node associated with a textual description $\mathcal{S} = \{s_1, ..., s_{|\mathcal{S}|}\}$, where $|\mathcal{S}|$ is the number of tokens in the textual description $\mathcal{S}$, and a partial set of node labels $C \subset \mathcal{V}$, the goal is to predict the labels of the remaining nodes $P = \mathcal{V} \setminus C$.

3.4 Methodology

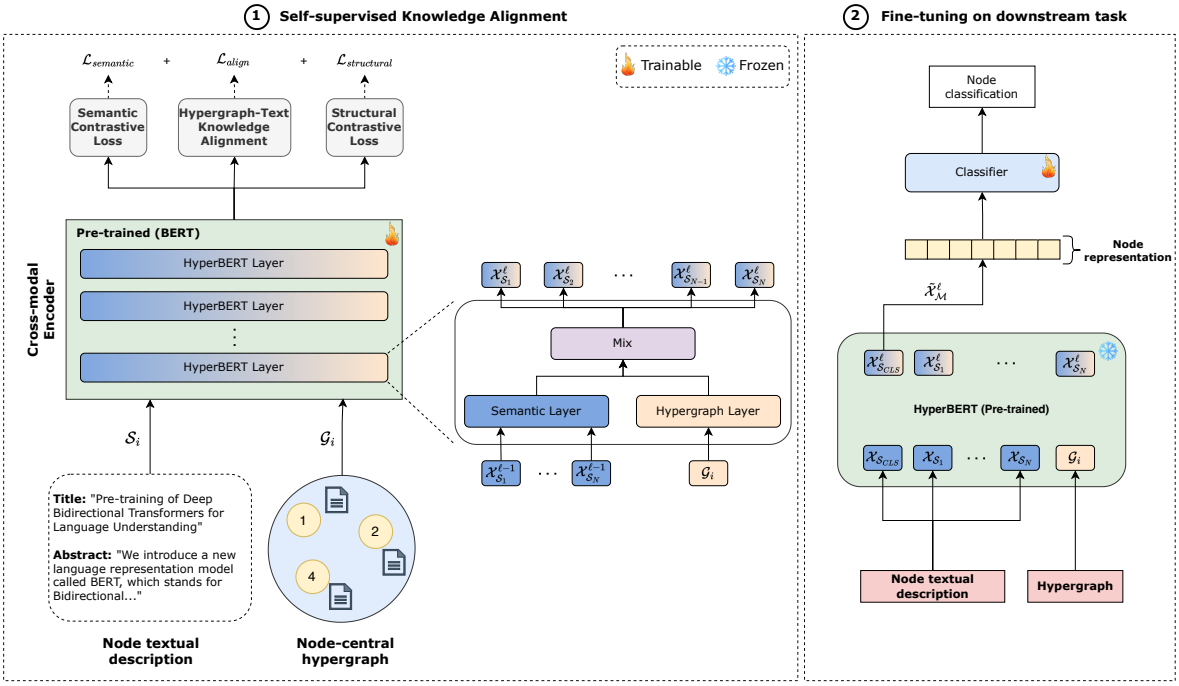

Figure 3.2: High-level overview of my proposed HyperBERT model. The model mixes hypergraph-aware layers into BERT to simultaneously exploit hypergraph topology and text semantics for node classification tasks on TAHGs. To accomplish the text-hypergraph alignment in the feature space, HyperBERT employs a novel self-supervised loss for pretraining that effectively aligns semantic and hypergraph feature spaces. After pretraining the model, it can be fine-tuned on a variety of downstream hypergraph node-level tasks such as node classification.

In this section I present my proposed model, HyperBERT, as illustrated in Figure 3.2. First, I begin with the details of the HyperBERT layer design. Then, I detail my proposed pre-training methodology, followed by details of the different components of my knowledge alignment loss function. Finally, I describe how fine-tuning is performed for a downstream node classification task.

3.4.1 The HyperBERT layer

The HyperBERT layer explicitly mixes semantic information obtained from each BERT Transformer block with structural information obtained with a Hypergraph Neural Network (HGNN) block.

Semantic Representation. The semantic representations of node textual attributes are encoded by a BERT [98] encoder. Specifically, BERT consists of alternating layers of multi-head self-attention (MHA) and Fully-connected Forward Network (FFN) blocks, as in the original Transformer [128] architecture. Before every block, Layer Normalisation (LN) is applied, and

after every block, a residual connection is added. More formally, in the ℓ^{th} encoder layer, the hidden states are represented as $\mathcal{X}_{\mathcal{S}}^{\ell} = \{x_1^{\ell}, \ldots, x_N^{\ell}\}$, where *N* is the maximum length of the inputs. First, a MHA block maps $\mathcal{X}$ into a query matrix $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$, key matrix $\mathbf{K} \in \mathbb{R}^{n \times d_k}$ and value matrix $\mathbf{V} \in \mathbb{R}^{n \times d_v}$, where *m* is the number of query vectors, and *n* the number of key/value vectors. Then, an attention vector is calculated as follows

$$\begin{aligned} \mathbf{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \mathbf{softmax}(\mathbf{A})\mathbf{V}, \\ \mathbf{A} &= \frac{\mathbf{Q}\mathbf{K}^{\mathbf{T}}}{\sqrt{\mathbf{d_k}}} \end{aligned} \tag{3.1}$$

In practice, the MHA block calculates the self-attention over *h* heads, where each head *i* is independently parametrised by $\mathbf{W_i^Q} \in \mathbb{R}^{d_m \times d_k}$, $\mathbf{W_i^K} \in \mathbb{R}^{d_m \times d_k}$ and $\mathbf{W_i^V} \in \mathbb{R}^{d_m \times d_v}$, mapping the input embeddings $\mathcal{X}$ into queries and key-value pairs. Then, the attention for each head is calculated and concatenated, as follows

$$\begin{aligned} \mathbf{Head_i} &= \mathbf{Attn}(\mathbf{Q}\mathbf{W_i^Q}, \mathbf{K}\mathbf{W_i^K}, \mathbf{V}\mathbf{W_i^V}) \\ \mathbf{MHA}(\mathcal{X}_{\mathcal{S}}^{\ell}) &= \mathbf{Concat}(\mathbf{Head_1}, \ldots, \mathrm{Head_h})\mathbf{W^U} \\ \bar{\mathcal{X}}_{\mathcal{S}}^{\ell} &= \mathbf{MHA}(\mathcal{X}_{\mathcal{S}}^{\ell}) \end{aligned} \tag{3.2}$$

where $\mathbf{W^U} \in \mathbb{R}^{d_m^h \times d_m}$ is a trainable parameter matrix. Next, to acquire the semantic hidden states of the input, a FFN block is applied, as follows

$$\mathbf{FFN}(\bar{\mathcal{X}}_{\mathcal{S}}^{\ell}) = \mathbf{max}(\mathbf{0}, \bar{\mathcal{X}}_{\mathcal{S}}^{\ell}\mathbf{W_1} + \mathbf{b_1})\mathbf{W_2} + \mathbf{b_2} \tag{3.3}$$

where $\mathbf{W_1} \in \mathbb{R}^{d_m \times d_{ff}}$ and $\mathbf{W_2} \in \mathbb{R}^{d_{ff} \times d_m}$ are linear weight matrices. Finally, layer normalisation and residual connection are applied as follows

$$\tilde{\mathcal{X}}_{\mathcal{S}}^{\ell} = \mathbf{LayerNorm}(\bar{\mathcal{X}}_{\mathcal{S}}^{\ell} + \mathbf{FFN}(\bar{\mathcal{X}}_{\mathcal{S}}^{\ell})) \tag{3.4}$$

Therefore, after the *L* encoder layers, I obtain the nodes' textual description semantic representations as the pooled feature tensor of the [CLS] token, denoted as $\tilde{\mathcal{X}}_{\mathcal{S}}^{\ell}$.

Hypergraph Structural Representation. In each HyperBERT layer, structural representations are produced through a Hypergraph Neural Network (HGNN) [78] over the node-centered context hypergraph $\mathcal{G}_i$, defined as the subhypergraph containing only the hyperedges which node i belongs to, and its incidence matrix H_i. Formally, I formulate the HGNN block to obtain hypergraph node embeddings as follows

$$\tilde{\mathcal{X}}_{\mathcal{G}}^{\ell} = \sigma(\mathbf{D}^{-1}\mathbf{H}\mathbf{W}\mathbf{B}^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{X}^{(\ell-1)}\mathbf{\Psi}), \tag{3.5}$$

where $\mathbf{D}$ and $\mathbf{B}$ are the degree matrices of the vertex and hyperedge in the hypergraph, respectively. $\sigma(\cdot)$ is a non-linear activation function such as ReLU [129]. $\mathbf{\Psi} \in \mathbb{R}^{d_\ell \times d_{(\ell-1)}}$ is a

learnable weight matrix between the ℓ^{th} and $(\ell-1)^{th}$ layer. Consequently, after *L* hypergraph layers, I obtain the nodes' structural representation as $\tilde{\mathcal{X}}_{\mathcal{G}}^{\ell}$.

Text-Hypergraph Joint Representation. After computing representations from both semantic and hypergraph structural spaces, the l^{th} HyperBERT layer computes a mixed text-hypergraph representation as follows

$$\tilde{\mathcal{X}}_{\mathcal{M}}^{\ell} = \tilde{\mathcal{X}}_{\mathcal{S}}^{\ell} + \tilde{\mathcal{X}}_{\mathcal{G}}^{\ell}, \tag{3.6}$$

where $\tilde{\mathcal{X}}_{\mathcal{M}}^{\ell}$ contains a mixture of semantic and hypergraph structural information to enable information flow between both modalities. Specifically, to combine structural and semantic embeddings, I add the hypergraph embedding to the embedding of each of the tokens in $\tilde{\mathcal{X}}_{\mathcal{S}}^{\ell}$. Therefore, $\tilde{\mathcal{X}}_{\mathcal{M}}^{\ell}$ effectively incorporates both the semantic information from text attributes and the structural context from the hypergraph. This is crucial for capturing the complex interplay between textual content and graph topology.

3.4.2 Hypergraph-aware pretraining task

In order to improve the learning capability of HyperBERT without using any human-annotated labels, I propose a novel hypergraph-aware knowledge alignment pretraining algorithm based on a contrastive learning loss. The proposed hypergraph-aware pretraining task exploits inherent hypergraph knowledge from the TAHG and is applied to both the semantic and structural representations.

Semantic Contrastive Loss. Based on the semantic representation of node i, $\tilde{\mathcal{X}}_{\mathcal{S}_i}$, the other nodes belonging to its same set of hyperedges, $\mathcal{N}(i)$, and the node i excluded mini-batch instances, $\mathcal{B}(i)$, the semantic contrastive loss objective can be defined as follows:

$$\mathcal{L}_{\texttt{semantic}} = \frac{-1}{|\mathcal{N}(i)|} \sum_{p \in \mathcal{N}(i)} \log \frac{e^{(\tilde{\mathcal{X}}_{\mathcal{S}_i} \cdot \tilde{\mathcal{X}}_{\mathcal{S}_p}/\tau)}}{\sum\limits_{j \in \mathcal{B}(i)} e^{(\tilde{\mathcal{X}}_{\mathcal{S}_i} \cdot \tilde{\mathcal{X}}_{\mathcal{S}_j}/\tau)}}, \tag{3.7}$$

where τ denotes the temperature and $\mathrm{e}(\cdot)$ represents the exponential function. Here for node i, I consider its semantic representation $\tilde{\mathcal{X}}_{\mathcal{S}_i}$ as the query instance. The positive instances are the representations of node i's hyperedge members $\{\tilde{\mathcal{X}}_{\mathcal{S}_p} \mid p \in \mathcal{N}(i)\}$. Meanwhile, the negative instances are the representations of other text nodes excluding i within the same mini-batch $\{\tilde{\mathcal{X}}_{\mathcal{S}_j} \mid j \in \mathcal{B}(i)\}$.

Structural Contrastive Loss. Similar to the semantic representation, I also apply my contrastive learning algorithm to the hypergraph structural feature space representation of node i, denoted as $\tilde{\mathcal{X}}_{\mathcal{G}_i}$. Formally, the structural contrastive loss is formulated as follows:

$$\mathcal{L}_{structural}=\frac{-1}{|\mathcal{N}(i)|}\sum_{p\in\mathcal{N}(i)}\log\frac{e^{(\tilde{\mathcal{X}}_{\mathcal{G}_i}\cdot\tilde{\mathcal{X}}_{\mathcal{G}_p}/\tau)}}{\sum\limits_{j\in\mathcal{B}(i)}e^{(\tilde{\mathcal{X}}_{\mathcal{G}_i}\cdot\tilde{\mathcal{X}}_{\mathcal{G}_j}/\tau)}}, \tag{3.8}$$

where $\tilde{\mathcal{X}}_{\mathcal{G}_i}$ is the query instance. The positive instances are $\{\tilde{\mathcal{X}}_{\mathcal{G}_p} \mid p \in \mathcal{N}(i)\}$ and the negative instances are $\{\tilde{\mathcal{X}}_{\mathcal{G}_j} \mid j \in \mathcal{B}(i)\}$.

Therefore, my semantic and structural contrastive learning losses ensure nodes belonging to the same hyperedges to share similar representations, which inherently elicits informative hypergraph knowledge during the learning process.

Hypergraph-Text Knowledge Alignment. In this work, my goal is to learn expressive representations that simultaneously encode informative textual semantics within each text node, as well as structural information among hyperedges. However, individually performing contrastive learning on either the semantic or structural spaces is not enough due to the lack of information exchange between both modalities. To better align the knowledge captured by the semantic and hypergraph structural layers, I propose a hypergraph-text knowledge alignment loss function for TAHGs.

In particular, for each node i, based on its semantic and structural representations, $\tilde{\mathcal{X}}_{\mathcal{S}_i}$ and $\tilde{\mathcal{X}}_{\mathcal{G}_i}$, respectively, I formulate the objective of hypergraph-text knowledge alignment loss, $\mathcal{L}_{align}$, as follows:

$$\begin{aligned}
\mathcal{L}_{align}&=\frac{-1}{|\mathcal{N}(i)|}\sum_{p\in\mathcal{N}(i)}\Big(\mathcal{L}_{align_1}+\mathcal{L}_{align_2}\Big)/2,\\
\mathcal{L}_{align_1}&=\log\frac{e^{(\tilde{\mathcal{X}}_{\mathcal{G}_i}\cdot\tilde{\mathcal{X}}_{\mathcal{S}_p}/\tau)}}{\sum\limits_{j\in\mathcal{B}(i)}e^{(\tilde{\mathcal{X}}_{\mathcal{G}_i}\cdot\tilde{\mathcal{X}}_{\mathcal{S}_j}/\tau)}},\\
\mathcal{L}_{align_2}&=\log\frac{e^{(\tilde{\mathcal{X}}_{\mathcal{S}_i}\cdot\tilde{\mathcal{X}}_{\mathcal{G}_p}/\tau)}}{\sum\limits_{j\in\mathcal{B}(i)}e^{(\tilde{\mathcal{X}}_{\mathcal{S}_i}\cdot\tilde{\mathcal{X}}_{\mathcal{G}_j}/\tau)}}
\end{aligned} \tag{3.9}$$

Given the above formulation, in the first component of the knowledge alignment loss function, $\mathcal{L}_{align_1}$, I treat the structural representation of node i, $\tilde{\mathcal{X}}_{\mathcal{G}_i}$, as the query, then construct the positive and negative instances based on the semantic representations. Specifically, the positive instances include both the representation of node i as well as the representations of i's same hyperedge nodes (i.e., $\{\tilde{\mathcal{X}}_{\mathcal{S}_p} \mid p \in \widetilde{\mathcal{N}}(i)\}$), and the negative instances are the representations of other instances within the same mini-batch $\{\tilde{\mathcal{X}}_{\mathcal{S}_j} \mid j \in \mathcal{B}(i)\}$. In the second component of the knowledge alignment loss, $\mathcal{L}_{align_2}$, I consider the semantic representation $\tilde{\mathcal{X}}_{\mathcal{S}_i}$ as the query and construct its corresponding positive and negative instances in the parallel way as for the first loss component. Consequently, the proposed $\mathcal{L}_{align}$ alignment loss encourages the representations of the same node learned across the two separate feature spaces, semantic and structural, to be

pulled together in the embedding space.

Learning Objective. In order to train my hypergraph-aware language model, I jointly optimise the proposed semantic, structural and hypergraph-text knowledge alignment losses. Specifically, the overall training loss objective is formulated as follows:

$$\mathcal{L}=\lambda_1\mathcal{L}_{semantic}+\lambda_2\mathcal{L}_{structural}+\lambda_3\mathcal{L}_{align}, \tag{3.10}$$

where λ_1, λ_2, $\lambda_3 \in \mathbb{R}$. In practice, I found that the values of $\lambda_1 = \lambda_2 = \lambda_3 = 1$ led to good experimental results.

3.4.3 Fine-tuning in downstream tasks

Once the pretraining is finished, I freeze the parameters of HyperBERT and use it to compute representations of each text node. In particular, during fine-tuning HyperBERT computes the text-hypergraph joint representation, $\tilde{\mathcal{X}}^{\ell}_{\mathcal{M}}$, capturing an aligned representation of the node's textual attributes and its topological structure in the hypergraph. Furthermore, this representation is used to fine-tune separate models on different downstream tasks. Specifically, I train a linear classifier from the node features to class labels for node classification using the cross entropy loss.

Method	Co-citation		Co-authorship		Movies
	Cora	Pubmed	DBLP-A	Cora-CA	IMDB
HGNN [78]	50.0 (7.2)	72.9 (5.0)	67.1 (6.0)	50.2 (5.7)	42.2 (2.9)
HyperGCN [115]	33.1 (10.2)	63.5 (14.4)	68.2 (14.4)	50.2 (5.7)	37.9 (4.5)
HNHN [130]	50.0 (7.9)	72.1 (5.4)	62.6 (4.8)	48.3 (6.2)	42.3 (3.4)
UniGCN [131]	49.1 (8.4)	74.4 (3.9)	65.1 (4.7)	51.3 (6.3)	41.6 (3.5)
UniGIN [131]	47.8 (7.7)	69.8 (5.6)	63.4 (5.1)	48.3 (6.1)	41.4 (2.7)
UniGCNII [131]	48.5 (7.4)	74.1 (3.9)	65.8 (3.9)	54.8 (7.5)	42.5 (3.9)
AllSetTransformer [132]	47.6 (4.2)	72.4 (4.5)	65.3 (3.9)	57.5 (5.7)	42.3 (2.4)
HyperGCL [133]	60.3 (7.4)	76.8 (3.7)	79.7 (3.8)	62.0 (5.1)	43.9 (3.6)
H-GD [134]	50.6 (8.2)	74.5 (3.5)	75.1 (3.6)	58.8 (6.2)	43.0 (3.3)
ED-HNN [135]	47.6 (7.7)	72.7 (4.7)	65.8 (4.8)	54.8 (5.4)	41.4 (3.0)
HypeBoy [117]	<u>62.3</u> (7.7)	<u>77.0</u> (3.4)	<u>80.6</u> (2.3)	<u>66.3</u> (4.6)	<u>47.6</u> (2.5)
HyperBERT	**64.5 (4.5)**	**78.9 (3.1)**	**82.3 (1.7)**	**69.2 (4.1)**	**49.7 (1.8)**

Table 3.1: Performance of HyperBERT on a variety of hypergraph node classification benchmarks. Cell values indicate mean and standard deviation accuracy calculated over 10 runs. Underscoring depicts the best performance for each dataset across previous methods. Bold indicates the best overall method for each dataset.

3.5 Experiments

In this section I show my model performance on five hypergraph node classification benchmarks from a variety of domains. Also, I present ablation studies to analyse the importance of the different components of the HyperBERT model.

3.5.1 Datasets and evaluation metrics

For my experiments I consider five datasets. The hypergraph datasets are from diverse domains and sizes, expressing co-citation, co-authorship and movie-actor relations. Following previous works on hypergraph node classification [117, 135], I calculate classification accuracy in all my experiments. I evaluate all models over 10 runs and report the mean and standard deviation of the test set accuracy averaged across them. Table 3.2 provides statistics on the datasets.

Dataset	Nodes	Hyperedges	# Classes
Cora	1,434	1,579	7
Pubmed	3,840	7,963	3
DBLP-A	2,591	2,690	6
Cora-CA	2,388	1,072	7
IMDB	3,939	2,015	3

Table 3.2: Statistics of the text-attributed hypergraph datasets used in my experiments.

I utilize two co-citation datasets: Cora and Pubmed. In these datasets, each node represents a publication and hyperedges represent sets of publications co-cited by particular publications. For example, if a publication has cited other nodes (publications) v_i, v_j and v_k, these nodes are grouped as a hyperedge $\{v_i, v_j, v_k\} \in \mathcal{E}$. Node classes indicate categories of the publications. Moreover, for co-authorship datasets I utilize Cora-CA and DBLP-A. In Cora-CA and DBLP-A, each node represents a publication, and a set of publications that are written by a particular author is grouped as a hyperedge. Classes indicate categories of the publication. I use IMDB for a movie-actor dataset. In this dataset, each node indicates a movie, and the movies that a particular actor has acted on are grouped as a hyperedge. Node attributes describe the plot of the movie and classes indicate the genre of the movie.

3.5.2 Implementation details

I implemented HyperBERT with PyTorch [136] and the HuggingFace library [137]. For the hypergraph neural network components, I use PyTorch Geometric [138]. For the knowledge

alignment stage, I set batch size as 32, and training steps to 20,000 on a single NVIDIA A100 GPU. I utilize the Adam optimiser with a fixed weight decay rate of 10^{-6} and learning rate of 10^{-3}. The number of layers for HyperBERT is set to 6, number of heads is 8 and dropout rate is 0.5. The dimensionality is set to 512. The temperature parameter τ contained in my loss, is set to 0.2. For training on downstream tasks, I use 200 epochs with an output MLP for node classification containing 2 layers and dimensionality of 512. I evaluate validation accuracy of the model every 10 epochs and keep the model checkpoint that yields the best validation accuracy, then I use it for predicting on the test dataset. Results are on the test set unless stated otherwise.

3.5.3 Overall performance

I first conducted experiments to evaluate model performance on text-attributed hypergraph node classification and present the results in Table 3.1. As shown in the table, my model HyperBERT outperforms all the baselines on the five hypergraph evaluation datasets, so demonstrating superior capability in text-attributed hypergraph node classification.

On the co-citation datasets, compared with AllSetTransformer [132], my model's classification accuracy is 64.5% for the Cora dataset, achieving 16.9% absolute performance improvement (47.6% vs 64.5%). In the case of PubMed, HyperBERT achieves 78.9% classification accuracy, which when compared with AllSetTransformer (72.4%) attains a 6.5% absolute accuracy increment. When compared with more recent methods that make use of self-supervised techniques for training the model, such as HypeBoy [117], my model's classification accuracy is 64.5% in Cora, a +2.2% improvement from 62.3%, and 78.9% in PubMed, a +1.9% increase from 77.0%. This effectively shows the benefit of my proposed architecture for solving text-attributed hypergraph node classification tasks.

In the co-authorship dataset experiments, when compared with the AllSetTransformer, HyperBERT attains an accuracy of 82.3% (+17%) and 69.2% (+11.7%) in the DBLP-A and Cora-CA datasets, respectively. In comparison with HypeBoy, HyperBERT obtains +1.7% and +2.9% performance improvements in the DBLP-A and Cora-CA datasets, respectively. Furthermore, my model reaches a performance in the IMDB movies dataset of 49.7%, an absolute improvement of 7.4% and 2.1% when compared with AllSetTransformer and HypeBoy, respectively.

3.5.4 Ablation studies

To investigate the contribution of each module in HyperBERT, I conduct several ablation studies to isolate and measure the impact of individual components on overall performance.

Effect of Architectural Components. In order to validate the architectural design of HyperBERT, I explore the effect of its different components towards node classification accuracy in the Pubmed dataset, and provide the results in Table 3.3. On the one hand, I investigate

suppressing particular components of the loss function, such as the semantic loss, structural loss, and alignment loss. To do so, I set their respective λ factors to 0, effectively cancelling their contribution during the pretraining stage. I find that these loss components lead to significant performance decreases when removed. Specifically, accuracy decreases to 66.4% (-12.5%), 68.8% (-10.1%) and 63.1% (-15.8%) when removing the semantic, structural and alignment losses, respectively. As expected, the biggest performance impact is given when eliminating the alignment loss, as it leads to completely disparate feature spaces for text and hypergraph node representations. I also study the impact of neglecting the high-order topology by replacing the HGNN encoder with a GNN, decreasing performance to 71.2% (-7.7%). In addition, I show the effect of not pretraining the model and instead training it from scratch on the downstream task, leading to a performance decrease of 71.5% (-7.4%). Therefore, the results of these experiments justify my architectural choices.

Method	Accuracy (%)
HyperBERT w/o semantic loss	66.4 ± 1.3
HyperBERT w/o structural loss	68.8 ± 1.6
HyperBERT w/o alignment loss	63.1 ± 2.3
HyperBERT with GNN as structural layer	71.2 ± 3.5
HyperBERT w/o pretraining	71.5 ± 1.5
HyperBERT	$\mathbf{78.9 \pm 3.1}$

Table 3.3: Effect of the different architectural components in node classification accuracy (and $\pm$ 95% confidence interval) of HyperBERT on the Pubmed benchmark.

Impact of Hypergraph Encoder. The choice of hypergraph encoder plays a crucial role in the computation of the hypergraph structural representations in HyperBERT. For this reason, Table 3.4 shows the node classification results in the PubMed dataset when using several different hypergraph architectures as hypergraph encoder layers while keeping BERT as the text encoder. As can be observed, the highest performance is achieved when using HGNN as the hypergraph encoder, with 78.9% ($\pm$ 3.1) accuracy, whereas the lowest is attained when using HyperGCN, leading to an accuracy of 76.3% ($\pm$ 2.5). These results highlight the significant impact that the choice of hypergraph encoder can have on model performance, suggesting that careful selection of the encoder architecture is essential for optimising node classification accuracy.

Encoder	Accuracy (%)
HyperGCN [115]	76.3 ± 2.5
HNHN [130]	76.9 ± 1.8
UniGCN [131]	77.6 ± 2.3
UniGIN [131]	77.5 ± 2.1
UniGCNII [131]	77.9 ± 3.4
ED-HNN [135]	78.2 ± 2.6
HGNN [78]	$\mathbf{78.9 \pm 3.1}$

Table 3.4: Accuracy of node classification on the PubMed test set using different hypergraph architectures as hypergraph encoder layers.

Impact of Text Encoder. The choice of text encoder is crucial in determining the overall performance of HyperBERT, particularly when the hypergraph encoder is fixed to HGNN. Table 3.5 presents the node classification results on the PubMed dataset using different text encoders while keeping HGNN as the hypergraph encoder. The highest performance is achieved with BERT, showing an accuracy of 78.9% ($\pm$ 3.1), while ELECTRA and RoBERTa demonstrate slightly lower accuracies of 78.4% ($\pm$ 2.9) and 78.2% ($\pm$ 3.0), respectively. These results indicate that while all three text encoders provide competitive performance, BERT remains the most effective choice in this setup.

Encoder	Accuracy (%)
RoBERTa [139]	78.2 ± 3.0
ELECTRA [140]	78.4 ± 2.9
BERT [98]	$\mathbf{78.9 \pm 3.1}$

Table 3.5: Accuracy of node classification on the PubMed test set using different text encoders while keeping HGNN as the hypergraph encoder.

3.6 Discussion and summary

In this chapter, I introduced HyperBERT, a novel hypergraph-aware language model that integrates hypergraph structural information with the semantic capabilities of BERT for the task of text-attributed hypergraph node classification. My approach is motivated by the need to simultaneously leverage rich textual attributes and the complex, higher-order connectivity inherent in hypergraphs, a combination that traditional graph learning methods have struggled to fully

exploit.

HyperBERT distinguishes itself by incorporating hypergraph-aware layers directly into the BERT architecture, thereby endowing the model with the ability to capture both local semantic details and global structural context. The design of HyperBERT was guided by several key insights. First, by embedding hypergraph topology into the learning process, my model is better positioned to identify subtle correlations and interactions among nodes that are only apparent when considering high-order relationships. Second, my choice of a self-supervised pretraining task based on contrastive learning facilitates effective alignment between the semantic and structural feature spaces. This alignment is crucial for enabling the model to generate robust and discriminative node representations.

Empirical evaluations on five benchmark datasets from diverse domains, spanning co-citation, co-authorship, and movie-actor relations, demonstrate that HyperBERT significantly outperforms existing baselines. The comprehensive ablation studies further underscore the importance of each component of my architecture. In particular, the semantic and structural contrastive losses, as well as the hypergraph-text alignment mechanism, each contribute to the overall performance, confirming my hypothesis that the joint modeling of text and hypergraph structure is essential for accurate node classification.

Overall, HyperBERT not only advances the state-of-the-art in text-attributed hypergraph node classification but also offers a promising framework for integrating multimodal data sources. The findings in this chapter highlight the potential of blending structural graph information with powerful language models, paving the way for future research in more efficient knowledge alignment and in exploring additional downstream tasks beyond node classification.

CHAPTER 4

WEAKLY SUPERVISED LANGUAGE MODEL SEMANTIC DISTILLATION FOR DENSE RETRIEVAL

4.1 Introduction and contribution overview

In recent years, the issue of automated claim verification has gained considerable attention as the volume of potentially misleading and false claims rises [141], resulting in the development of fully automated methods for claim checking (see [141–145] for recent surveys). Pioneering research in the field of Natural Language Processing (NLP) has led to the emergence of (large) language models (LMs) (e.g. [42, 99–101]). These models have been successful in many applications due to the vast implicit knowledge contained within them, and their strong capabilities for semantic understanding of language. However, issues around fact hallucination have gained considerable attention [146, 147] and are a major concern in the widespread usage of LLM-based applications across different settings.

As the world becomes more aware of the issues around information trustworthiness, the importance of developing robust claim verification techniques grows ever more critical. Historically, the design of claim verification methods has been enabled by the creation of annotated datasets, such as FEVER [142] or MultiFC [148], of appropriate scale, quality, and complexity in order to develop and evaluate models for claim checking. Most recent methods for this task have been dominated by two approaches: natural language inference (NLI) models (e.g., [142, 149–153]), and knowledge graph-augmented methods (e.g. [70, 154–157]). These proposals mainly leverage the NLI methods to model the semantic relationship between claim and evidence, or further make use of the knowledge graph (KG) structure to capture the features underlying between multiple pieces of evidence.

However, these studies have largely relied on annotated data for model training, and while

gathering data is often not difficult, its labelling or annotation is always time-consuming and costly. Thus, an emerging trend in the literature [158–161] is to move away from annotation-dependant methods and try to learn patterns in the data using unsupervised or weakly supervised training methods. With the advent of such training methods, new avenues have opened for research into leveraging the huge amounts of unlabelled data to achieve better performance more efficiently. Despite significant advancements in the field of weakly supervised learning, only a handful of strategies have been proposed for textual claim verification (e.g. [162–165]). Thus there are still opportunities for the development of weakly supervised techniques tailored specifically for such tasks.

Following recent trends in weakly supervised methods, I eliminate data annotation requirements and instead, without human supervision, automatically try to identify relevant evidence for fact-checking. Thus, in this chapter I present a novel weakly supervised feature representation learning strategy with well-designed sub-tasks for automatic claim-fact matching. My method leverages pre-trained features from Language Models and focus on distilling them into compact and discrete structures that attain a high alignment between the textual claims to be verified, and their corresponding evidence in the knowledge graph. In particular, my contributions are summarized as follows:

- I introduce a novel weakly supervised training method, applied specifically to the task of claim verification on textual claims and knowledge graph-based evidence by language model distillation.
- I demonstrate that my method achieves state-of-the-art performance in claim verification and knowledge retrieval benchmarks, such as FEVER and FB15k-237.
- I justify my architectural design decisions with ablation studies on the main components.

4.2 Related work

Claim verification with pre-trained language models. Most recent works typically divide the claim verification task into two stages. The first stage retrieves a relatively small subset of evidence from a knowledge source (e.g. a knowledge graph) that is relevant to verify a given claim. The second stage performs reasoning over the retrieved evidence to discern the veracity of the claim. Such retrieval-and-reasoning approaches aim to reduce the search space, and have proven their superiority over directly reasoning on the whole knowledge graph [166, 167]. In order to match evidence with claims, a typical approach is to devise claim-fact similarities using semantic matching with neural networks. Due to the great semantic understanding recently demonstrated by pre-trained language models (PLMs), some recent works employ PLMs for addressing the claim-fact semantic matching task. In this vein, some works exploit the implicit knowledge stored within LMs for performing zero-shot claim checking, without any external

knowledge or explicit evidence retrieval [168, 169]. However, such methods are prone to suffer from hallucination errors, resulting in incorrect predictions. Other work, such as ReAct [170], explores the use of LMs to generate both reasoning traces and task-specific actions over a knowledge base by in-context learning via prompting, overcoming prevalent hallucination issues by interacting with a Wikipedia API. However such an approach is limited by input length making it impractical in complex tasks. In my method I distill the features of recent pre-trained language models to yield highly-correlated claim and evidence embeddings. I make use of a set of eight language models as backbones because of their quality, but note that my method can work with any language model.

Unsupervised and weakly supervised training methods for claim verification. Learning meaningful features for claim-fact matching without human labels is a nascent research direction in claim verification approaches, with recent works relying on self-supervised techniques. For instance, CosG [171] proposes a graph contrastive learning approach to learn distinctive representations for semantically similar claims with differing labels, with the goal of mitigating the over-smoothing issues commonly found in graph-based approaches. The model incorporates both unsupervised and supervised contrastive learning tasks to train a graph convolutional encoder, enhancing the representation of claim-fact pairs in the embedding space. [172] presents SSDL, a multi-task learning strategy that initially builds a student classifier using both self-supervised and semi-supervised methods, and then fine-tunes the classifier using distilled guidance from a larger teacher network that remains frozen during training. However, this method requires pairs of text claims and corresponding visual information as training data. [173] introduces KEGA, a knowledge-enhanced graph attention network for claim verification, which uses external knowledge bases to improve claim and evidence representations. It uses a contrastive learning loss to capture graph structure features. With BERT as its backbone, the model includes knowledge from WordNet and pre-trained knowledge base embeddings to enrich token representations, while a graph attention network [174] and the contrastive loss further enhance the model's ability to reason. LaPraDoR [175] introduces a pre-trained dense retriever approach with contrastive learning for unsupervised training of query and document encoders. The method is applied in a variety of text retrieval challenges, with FEVER being one of them. However, it shows a significant performance gap when compared against the supervised state-of-the-art approaches for FEVER. One of the main reasons is the lack of task-specific contrastive functions. In contrast to these works, my method is designed with a task-specific weakly supervised feature representation strategy, leveraging language model distillation to achieve high-quality claim-fact weakly supervised matching on large scale datasets for claim verification.

Knowledge distillation. Knowledge distillation seeks to transfer the knowledge from a (usually large) model, called the teacher, to another (usually small) model, called the student. This technique is often used for increasing the performance of the small model. One of the first

approaches for knowledge distillation was proposed by [176], via minimising the KL-divergence between the teacher and student's logits, using the predicted class probabilities from the teacher as soft labels to guide the student model. Instead of imitating the teacher's logits, [177] distilled knowledge by minimising the $\mathbb{L}_2$ distance between the intermediate outputs of the student and teacher. [178] aligned the pair-wise similarity graph of the student with the teacher, and [179] used the attention map generated by the teacher to force the student to attend to the same areas as the teacher. More recently, knowledge distillation has been extended for self-supervised settings. For instance, [180] use the contrastive loss to enforce cross-modality consistency. [181] and [182] aim at aligning the features between views of the same instances by computing pair-wise similarities between the student's outputs and features kept in a feature memory bank produced by the teacher. In this work, I propose to transfer the semantic knowledge of a pre-trained language model within a new type of self-supervised task, claim verification, by leveraging the language model capabilities of language understanding to guide the student to produce high-quality features for claim-fact matching.

4.3 Methodology

In this section I present my proposed weakly supervised semantic distillation training approach in detail, as illustrated in Figure 4.1a. First, I begin with the data processing pipeline. Then, I detail my proposed pre-training methodology, followed by details of the different components of my proposed contrastive loss function. Finally, I describe the optional adaptation phase, where I fine-tune the model pre-trained with my framework for a downstream claim verification classification task.

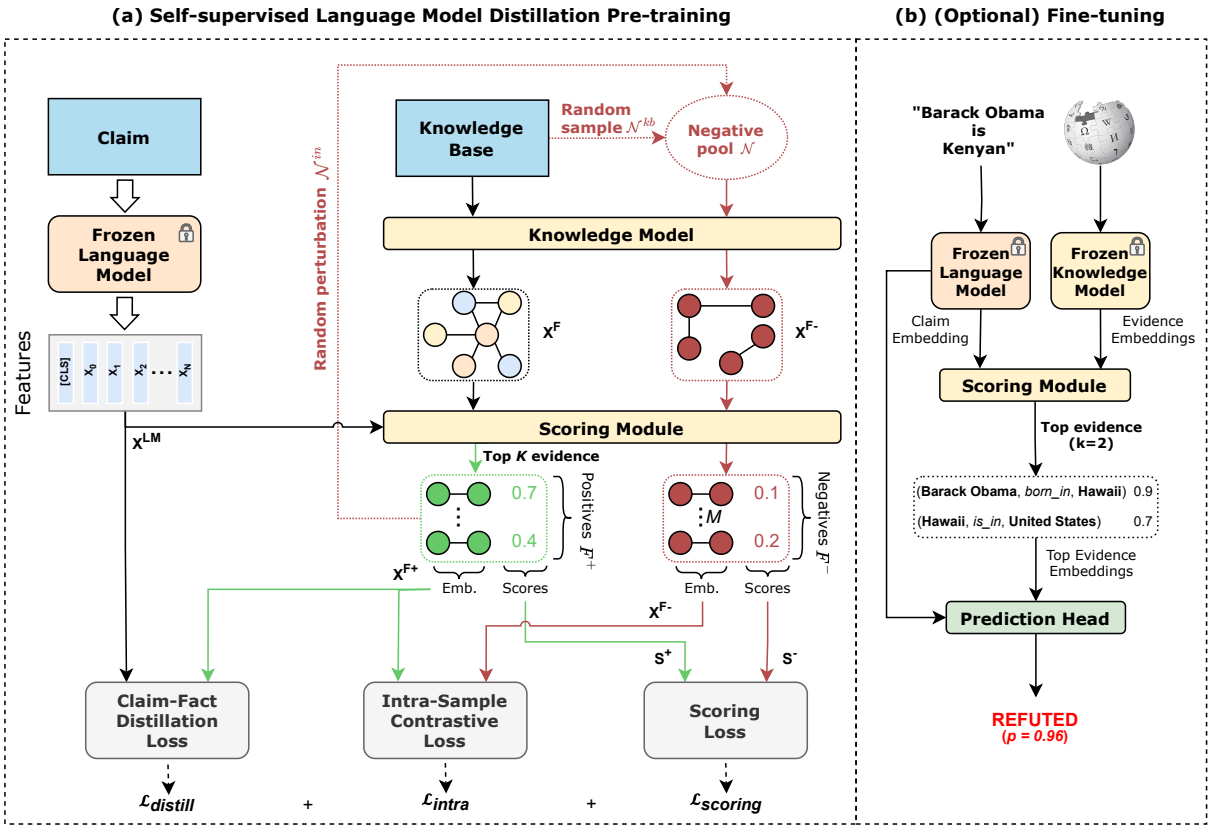

Figure 4.1: (a) A high-level overview of my training framework. Given a textual claim, I use a frozen language model (orange box) to obtain its embedding features, X^{LM}. The knowledge base is fed to the knowledge model to produce a knowledge base embedding X^F. Then, the scoring module produces scores for facts in the knowledge base, conditioned upon the claim embedding. The positive sub-graph formed by the top K facts is kept, denoted as X^{F^+}. Next, a negative pool of instances $\mathcal{N}$. Finally, both the positive and negative sub-graphs are encoded with the knowledge model, obtaining the positive and negative sub-graph embeddings, X^{F^+} and X^{F^-}, and their respective scores, S^+ and S^-. Grey boxes represent three the different components of my self-supervised loss function used to train the knowledge model. (b) Optional supervised fine-tuning stage on a downstream task using the pre-trained model.

4.3.1 Data processing pipeline

Throughout this section, I assume to have sampled a batch of unlabelled claims $x = \{x_i\}_{i=1}^{N}$, with x_i being the i^{th} claim and N the batch size. In addition, I assume access to a knowledge base represented as a knowledge graph $G(\varepsilon, \mathcal{R})$, where ε, $\mathcal{R}$ are the set of real-world entities associated with a name (e.g. Barack Obama, New York City), and the relationship type between entities (e.g. $was\ born\ in$), respectively. I also define F as the set of facts contained in G, represented as triples of subject, relation, object, named as head, relation and tail. Then, G can be defined as $G = \{(h_\mathrm{i}, r_\mathrm{i}, t_\mathrm{i}) \mid h_\mathrm{i}, t_\mathrm{i} \in \varepsilon, r_\mathrm{i} \in \mathcal{R}\}$, where $i \in \{1, \ldots, |\mathrm{F}|\}$.

4.3.2 Pretraining method

As shown in Figure 4.1a, my pre-training approach uses a pre-trained language model to obtain a feature tensor for each of the input claims. In my method, I use a knowledge model to embed

facts from the knowledge graph, and a scoring module that scores each of such facts conditioned upon a specific claim. To tackle the discrepancy in information-level between the features from the pre-trained language model and the knowledge model, I introduce a weakly supervised distillation loss that encourages the representation of the claim and its related knowledge facts to be mapped close together in the feature space of the knowledge model. A scoring loss function encourages the scoring module to provide higher scores for positive facts than to randomly-generated negative facts. To avoid the network finding trivial solutions where both positive and negative facts are given similar scores, a contrastive loss is used to encourage the separation, within the feature space, of features representing the positive and negative facts.

First, the knowledge model is initialized randomly and the backbone is initialized from any off-the-shelf pre-trained language model (e.g. a T5; [183]). In this work, the backbone f_L is kept frozen during the entire training and is only used to obtain a feature tensor for each claim, denoted as X^{LM}, which is used for distillation on the much smaller knowledge model. In order to obtain a single tensor representation per claim, I take the global average pooling (GAP) of the backbone features for each claim. For the knowledge model, I utilize a Relational Graph Attention Network [184].

Next, the knowledge model, $f_G : G \rightarrow \mathbb{R}^{|V| \times d_{\text{V}}}$, maps the input knowledge graph, G, into $X^{KB} \in \mathbb{R}^{|V| \times d_{\text{V}}}$, where $|V|$ is the number of nodes in G and d_{V} is the feature space dimensionality. In order to obtain a single feature tensor for each fact in F, I use a multilayer perceptron (MLP) that combines the head and tail embeddings for each fact into a single tensor, denoted as $X^F \in \mathbb{R}^{|F| \times d_{\text{F}}}$.

Then, given a single claim and fact embedding, denoted as X_i^{LM} and X_i^F, respectively, I propose a score function, f_{score}. The goal of f_{score} is to measure how likely it is that a given fact is in the same context as the corresponding claim. Specifically, the calculation of f_{score} is defined as:

$$f_{\text{score}}(X_i^{LM}, X_i^F) = d(X_i^{LM}, X_i^F) \tag{4.1}$$

where $d(.)$ is a similarity score, which I take to be the $\mathbb{L}_2$ norm. Therefore, given the embedding of claim x_i, and every fact embedding in the knowledge base, X^F, I can compute the relevance scores $S_i = \{f_{\text{score}}(X_i^{LM}, X_j^F) \mid \forall\, \text{j} \in \{1, \ldots, |\text{F}|\}$. Then, the set of most relevant facts corresponding to claim x_i is defined as:

$$F_i^+ = \text{top-rank}(S_i, K) \tag{4.2}$$

where top-rank($\cdot$, K) returns the indices of top K items in a set. I can now obtain the embeddings of the positive facts, $X^{F^+} \subset X^F$, and the corresponding scores, $S^+ \subset S$, by using the indices of the top K facts according to the scores S.

4.3.3 Generation of negative instances

In Section 4.3.2 I have described the process of obtaining both the positive fact embeddings and scores, X^{F^+} and S^+, respectively. In this section I explain how to harness the graph structure of the knowledge base to produce corresponding negative signals for contrastive learning.

In order to produce a negative set for claim x_i, herein denoted as $\mathcal{N}_i$, I take inspiration from recent advances in graph contrastive learning (e.g. [185–187]), and propose to generate two sets of negative instances: in-batch negatives and in-knowledge-base negatives, denoted as $\mathcal{N}_i^{in}$ and $\mathcal{N}_i^{kb}$, respectively. My approach aims to generate negative samples that are factually false while preserving the contextual meaning of the entities, so that meaningful negative samples are fetched.

In-batch negatives. To generate in-batch negatives I perform a random perturbation of the entities in the set of positive facts F_i^+ for a given claim x_i. Formally, let us define the set of triples in the positive set of claim i as $T_i = \{(h_\text{i}, r_\text{i}, t_\text{i}) \mid \forall\, \text{i} \in \{1, 2, \ldots, |F_i^+|\}$, where h, r, t represents head, relation and tail, respectively. My goal is to generate $\mathcal{M}$ negative samples in each given batch $\mathcal{B}$. For each triple $t_{h,r,t}$ in T_i, I decide in a probabilistic manner whether the perturbation is done on the head or the tail of the triple. For this, let us define a random variable $perturb\_head$ $\sim$ Bern(p_{head}) sampled from a Bernoulli distribution with parameter p_{head}, dictating whether the head of a triple should be perturbed, with probability p_{head}, or the tail otherwise. Then for each triple $t_{h,r,t}$, I generate a negative triple $t_{h',r,t}$ or $t_{h,r,t'}$, by altering head (p_{head} = 1) or tail (p_{head} = 0), respectively, such that the new head, $h^{'}$, or tail, $t^{'}$, are sampled from ε uniformly. To provide semantically meaningful negatives, I enforce the entity type of the randomly sampled head/tail to be of the same type as the one in $t_{h,r,t}$.

In-knowledge-base negatives. Given the nature of the in-batch negative generation process, the negative triples are bounded to be semantically similar to the corresponding positive triples, and hence close by in the feature space. Therefore, this bias leads to under-exploration of other parts of the knowledge base feature space. In order to alleviate this issue, I propose to add randomly-sampled facts from the knowledge base into the negative set. Specifically, given the knowledge base G, I sample $\mathcal{M}$ triples that are at least H hops away from F_i^+. This encourages the negative generation procedure to dynamically explore other parts of the knowledge base.

To obtain the final negative set for claim x_i, I join both the set of in-batch negatives $\mathcal{N}_i^{in}$ and in-knowledge base negatives $\mathcal{N}_i^{kb}$ as:

$$\mathcal{N}_i = \mathcal{N}_i^{in} \cup \mathcal{N}_i^{kb} \tag{4.3}$$

Finally, I obtain the embeddings of the negative set, X^{F^-} from the knowledge model, and the negative scores from the scoring module as $S^- = \{f_\text{score}(X_i^{LM}, X_j^{F^-}) \mid \forall\, \text{j} \in \{1, 2, \ldots, |N_i|\}$.

4.3.4 Language model distillation for semantic matching

Once I get positive and negative fact embeddings and scores, they can be used for the distillation process. In particular, I seek to learn a low-dimensional embedding that "distils" the feature correspondences of a pre-trained language model between textual claims and the knowledge base features produced by the knowledge model. To achieve this, I propose a loss function composed of 3 terms: claim-fact distillation, $\mathcal{L}_{distill}$, intra-sample contrastive loss, $\mathcal{L}_{intra}$, and scoring loss, $\mathcal{L}_{scoring}$.

Claim-Fact Distillation. To transfer the feature correspondences from the pre-trained language model to the knowledge model, I propose a feature-based [179] claim-fact distillation loss. Specifically, I propose the following distillation loss:

$$\mathcal{L}_{distill} = \sum_{j \in F^+} \left|\left| \frac{F_j^{KM}}{\|F_j^{KM}\|_2} - \frac{F_j^{LM}}{\|F_j^{LM}\|_2} \right|\right|_p^p. \tag{4.4}$$

where F_j^{KM} and F_j^{LM} are respectively the knowledge model (student) and language model (teacher) feature representations, for each fact j, in the positive fact set, F^+. p refers to the norm type, and I use $p = 2$ for $\mathbb{L}_2$-normalized features. To obtain F_j^{LM}, I compute the pooled features representation of the verbalized triplet j given by the LM, similarly to [188]. Therefore, the distillation is computed as the Mean Squared Error on the normalized fact embeddings.

Intra-Sample Contrastive Loss. The intra-sample distillation loss derives from the standard contrastive loss. The aim of the contrastive loss is to learn representations by discriminating the positive instance, among negative samples. For instance, in MoCo [161], two views, x and $x^{'}$, of one input image are obtained using augmentation, and an encoder f_q and momentum encoder f_k are used to generate embeddings of the positive pairs, such that $q = f_q(x)$ and $k = f_k(x^{'})$. In this case, the contrastive loss can be defined as:

$$\mathcal{L}_{contrastive} = -\log \frac{\exp(\mathbf{q} \cdot \mathbf{k}^+/\tau)}{\sum_{i \in N} \exp(\mathbf{q} \cdot \mathbf{k_i}/\tau)}. \tag{4.5}$$

Similarly to [189], I extend the contrastive loss function by replacing the query, q, in the original formulation with the claim embedding, denoted as X_i^{LM}, and extending the formulation to account for multiple positive keys. Consequently, I contrast the query with respect to each of the individual positive (numerator) and negative (denominator) facts, as follows:

$$\mathcal{L}_{intra} = \frac{-1}{|F^+|} \sum_{i \in F^+} \log \frac{\exp(X_i^{LM} \cdot X_i^{F^+}/\tau)}{\sum_{j \in F^-} \exp(X_i^{LM} \cdot X_j^{F^-}/\tau)}. \tag{4.6}$$

where τ is the temperature parameter, used during training to smooth the logits distribution. The rationale of $\mathcal{L}_{intra}$ is to pull the positive fact embeddings close to the claim representation

while simultaneously pushing away the negative fact embeddings.

Scoring Loss. The scoring loss is a variant of the conventional pair-wise ranking loss [190]. Ranking losses are used to evaluate the performance of a learned ranking function. In this work, I propose the $\mathcal{L}_{scoring}$ loss function to maximise the scores given by the scoring model to the positive facts, F^+, and minimise the scores of the negative facts, F^-, for a given claim x_i. In particular, I minimise the following loss:

$$\mathcal{L}_{scoring} = \sum_{i \in F^+} \sum_{j \in F^-} max\left(0, \gamma + f_{\text{score}}(X_i^{LM}, X_i^{F^+}) - f_{\text{score}}(X_i^{LM}, X_j^{F^-})\right) \tag{4.7}$$

where γ is a margin factor. Minimising $\mathcal{L}_{scoring}$ encourages elements of $f_{\text{score}}(X_i^{LM}, X_i^{F^+})$ to be highly ranked and elements of $f_{\text{score}}(X_i^{LM}, X_j^{F^-})$ to have low scores. More explicitly, it optimises the model parameters so that the scores of positive facts, $(h, r, t) \in F^+$, are higher than the scores of negative facts $(h', r', t') \in F^-$.

Finally, the full loss is formalized as follows:

$$\mathcal{L}_{total} = \lambda_{distill}\mathcal{L}_{distill} + \lambda_{intra}\mathcal{L}_{intra} + \lambda_{scoring}\mathcal{L}_{scoring} \tag{4.8}$$

where $\lambda_{distill}$, λ_{intra}, $\lambda_{scoring} \in \mathbb{R}$. In practice, I found that a ratio of $\lambda_{intra} \approx \lambda_{scoring} \approx 2\lambda_{distill}$ led to good experimental results.

4.4 Experiments

In this section, I present a comparative study of the results of my proposed method on standard benchmarks for claim verification, as well as ablation studies on the most relevant components. I first describe the datasets, evaluation and training settings. Next, I discuss extensive experiments on my method for the task of claim verification. Finally, I run a set of ablation studies to evaluate the impact of the most important components of my proposed framework.

4.4.1 Implementation details

Datasets and evaluation. I use the FEVER [142] dataset for all my experiments and comparison against previous methods. For pre-training I use the official FEVER training set. For providing the performance comparisons against previous work, I use the official FEVER testing set. In my ablation studies, I employ the official FEVER development split. To evaluate the learning performance in a low-data regime, I randomly sample 1%, 5% or 10% of the training data. As knowledge base, I use the Wikidata5m [191]. I provide some examples of claims from FEVER in Table 4.1. Finally, I also compare on the FB15k-237 dataset from [192].

Claim	Evidence	Label
The Rodney King riots took place in the most populous county in the USA	(1992 Los Angeles riots, also_known_as, Rodney King riots), (Rodney King riots, occurred_in, Los Angeles County), (Los Angeles County, county_of, USA), (Los Angeles County, most_populous_county_of, USA)	SUPPORTED
Giada at Home was only available on DVD	(Giada at Home, is_a, Televion Show), (Giada at Home, aired_on, Food Network), (Food Network, is_a, Television Channel)	REFUTED
Al Jardine is an American rhythm guitarist	(Alan Charles Jardine, is_a, guitarist), (Al Jardine, alias_of, Alan Charles Jardine), (Alan Charles Jardine, has_nationality, American)	SUPPORTED
Bob Ross created ABC drama The Joy of Painting	(Robert Norman Ross, also_known_as, Bob Ross), (Robert Norman Ross, is_a, Painter), (Robert Norman Ross, is_a, Television host), (Robert Norman Ross, creator_of, The Joy of Painting), (The Joy of Painting, is_a, Instructional television program)	REFUTED

Table 4.1: Examples from FEVER [142] of claims, their annotated evidence and label.

Pretraining. Eight models with a variety of sizes are used as pre-trained language models: T5-Small [183], DeBERTaV3 [193], XLNet [194], RoBERTa [139], BERT [98], Transformer-XL [195] and GPT-2 [101]. The pre-trained language models are kept frozen during pre-training with my method. The officially released weights from HuggingFace [196] are used to initialize the pre-trained language models for fair comparisons. Pre-training is run for a total of 1000 epochs. During training I use a RGAT as the knowledge model with 3 convolutional layers, with a hidden size of 512. The projector from node embeddings to triple embeddings is a MLP with the same dimensionality as the pre-trained language model sentence embedding size. The model is trained with the SGD optimiser with momentum 0.9 and weight decay 0.0001. The batch size is set to 512 over 4 A100 GPUs, and the coefficients for the different losses are $\lambda_{intra} = \lambda_{scoring} = 1$, $\lambda_{distill} = 2$. I set the temperature τ = 0.1. I use K = 5 for the number of facts to keep after scoring. The number of negative instances used in the negative pool for contrastive learning is set to M = 4096.

Linear Probe Fine-tuning. In order to evaluate the quality of the distilled claim-fact matching features, I follow common evaluation protocols [197, 198] for measuring transfer learning effectiveness. Specifically, I train a linear classifier to perform label assignment to claims (see Figure 4.1b for an example illustration). The classifier is trained for 200 epochs, using the SGD optimiser with 20 as the initial learning rate. The only purpose of this linear probe is to evaluate the quality of the features and is not part of the pretraining procedure.

4.4.2 Experiments on the FEVER dataset

I summarize my results on the FEVER claim verification benchmark in Table 4.2. My method significantly outperforms prior state of the art. In particular, I improve by +9.23% on label accuracy in the test set when using a simple linear probe and a frozen backbone pre-trained using my method. Notably, even though my method has been trained without any data annotations, it is capable of outperforming the best supervised method (ProoFVer) by +9.58% label accuracy. These experiments demonstrate the benefits of my task-specific weakly supervised training framework for learning rich feature representations for claim-fact matching.

Method	**Unsupervised**	**Dev**		**Test**	
		LA	**Fever Score**	**LA**	**Fever Score**
GEAR [70]	✗	74.84	70.69	71.60	67.10
KGAT [157]	✗	78.29	76.11	74.07	70.38
[199]	✓	<u>81.21</u>	-	74.39	-
GERE [200]	✗	79.44	77.38	75.24	71.17
CorefBERT [153]	✗	-	-	75.96	72.30
DREAM [154]	✗	79.16	-	76.85	70.60
ProoFVer [201]	✗	80.74	<u>79.07</u>	79.47	<u>76.82</u>
[162]	✓	80.20	-	<u>80.25</u>	-
SFAVEL	✓	**90.32**	**88.10**	**89.48**	**86.15**

Table 4.2: Performance on the FEVER benchmark (label accuracy and FEVER score in %) of my proposed pre-training approach after fine-tuning a linear classification probe on the FEVER benchmark. In this experiment I use the GPT-2-XL as backbone. Underlined performances indicate the top score for a particular metric, bold indicates the overall best method. Dataset splits are from [142].

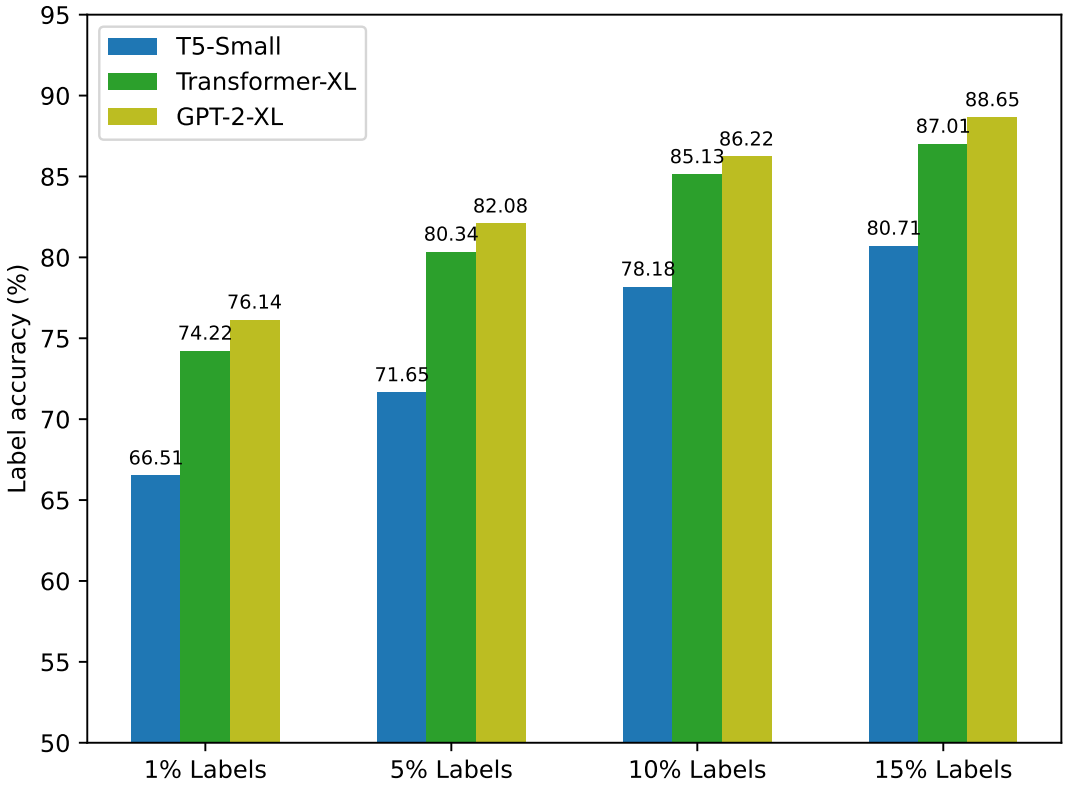

Figure 4.2: Low-data experiments by fine-tuning on 1%, 5%, 10% and 15% of data over 3 different language backbones.

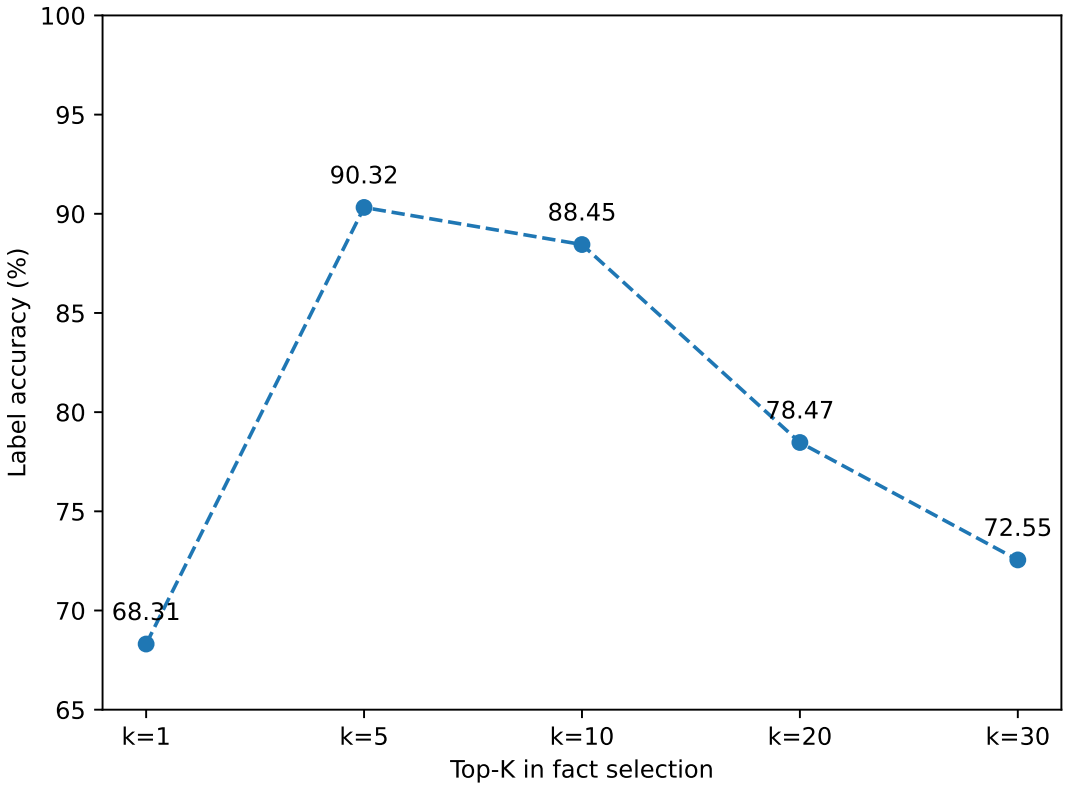

Figure 4.3: Label accuracy with different K for number of facts selected after scoring.

Furthermore, following previous works in contrastive learning [202], I evaluate the proposed method by distilling 3 different language model backbones (T5-Small, Transformer-XL, GPT-2-XL) and fine-tuning in a low-data setting by using 1%, 5%, 10% and 15% of labelled data. As shown in Figure 4.2, my method is capable of achieve on-par performance with recent methods despite only fine-tuning with 1% of the data, reaching 74.22% and 76.14% dev set accuracy with Transformer-XL and GPT-2-XL backbones, respectively. When using 5% of labelled data, my method surpasses previous state-of-the-art on the FEVER benchmark. This experiment highlights the high-quality features my framework is capable of learning for claim-fact matching, allowing high accuracy even when only a few labelled data points are available.

4.4.3 Experiments on the FB15k-237 dataset

In addition to my evaluations in Section 4.4.2 I compare my method to prior work on the FB15k-237 claim checking task presented in [192]. I utilize the same evaluation metrics and dataset processing as [203]. In this setting, in contrast to FEVER evaluation, I obtain the top K facts given a claim directly from my weakly supervised pre-trained model, without the need for a downstream prediction head. In Table 4.3 I find that my method is able to achieve +5.3%, +3.6% and +6.3% for Hits@1, Hits@3 and Hits@10, respectively, compared to the previous state of the art.

Method	Hits@1	Hits@3	Hits@10
KG-BERT [127]	-	-	42.0
StAR [204]	20.5	32.2	48.2
KG-S2S [205]	25.7	37.3	49.8
SimKGC [206]	24.6	36.2	51.0
MoCoSA [203]	29.2	42.0	57.8
SFAVEL	**34.5**	**45.6**	**64.1**

Table 4.3: Performance on the FB15k-237 dataset test split. I use the GPT-2-XL as backbone for distillation. Following previous methods, I measure performance using the Hits@1, 3, 10 metric. I obtain the top K facts from my weakly supervised pre-trained model, without the need for a downstream prediction head.

4.4.4 Ablation studies

In the following section, I provide several ablation studies for my proposed approach. All experiments and results are performed on the FEVER development set with the GPT-2-XL as language model backbone unless explicitly stated.

Pre-trained Language Model. To understand the impact of pre-trained language model selection for distillation backbone, I perform an ablation study and report the results in Table 4.4. I analyze the effect of using several different language models during the weakly supervised training stage, such as T5-Small, DeBERTaV3, XLNet, GPT-2-Small, RoBERTa, BERT, Transformer-XL and GPT-2-XL. I choose this particular set of language models as they are diverse in terms of their number of parameters. The smallest language model in my experiments is T5-Small (60 Million parameters), with the biggest LM being GPT-2-XL (1.5 Billion parameters). This gives some insight into how the language representation capabilities of each of the models affects the distillation effectiveness. I find that the GPT-2-XL is the best feature extractor of the list and leads by a significant margin in terms of accuracy. However, I note that even the smallest backbone (T5-Small; 60M parameters), although modestly, achieves performance greater than the previous state-of-the-art (+0.54% accuracy).

Backbone	# Params	Accuracy
T5-Small	60M	80.79
DeBERTaV3	86M	82.09
XLNet	110M	84.46
GPT-2-Small	117M	84.92
RoBERTa	125M	85.31
BERT	140M	87.92
Transformer-XL	257M	89.51
GPT-2-XL	1.5B	**90.32**

Table 4.4: Accuracy of linear classification results on FEVER dev set using different pre-trained language models as backbone for distillation.

Backbone	Distill Loss	Cont Loss	Scoring Loss	Accuracy
T5-Small		✓	✓	56.34
T5-Small	✓		✓	72.71
T5-Small	✓	✓		77.46
T5-Small	✓	✓	✓	80.79
GPT-2-XL		✓	✓	57.20
GPT-2-XL	✓		✓	79.64
GPT-2-XL	✓	✓		87.16
GPT-2-XL	✓	✓	✓	**90.32**

Table 4.5: Effect of the different components of my loss function. From left to right: language model backbone used, whether claim-fact distillation, intra-sample contrastive and scoring losses are used, respectively.

Influence of K in fact selection. I inspect the impact of K in fact selection after scoring for model performance. As shown in Figure 4.3, the results are consistent for a range of K (K = 1, 5, 10, 20, 30). In particular, I observe a decrease in classification accuracy with K = 10, 20, 30 compared with K = 5. I believe this decrease is caused by the factual noise introduced when K becomes large, where irrelevant information is used for verifying the specific claim. In contrast, with K = 1, the performance drop is caused by a lack of information, as only a single fact is used to check a claim. Avoiding this is critical in settings where multiple pieces of evidence are required for reasoning, as it is for FEVER.

Loss function components. I evaluate the different loss functions described in Section 4.3.4, and provide the results in Table 4.5. In particular, I investigate suppressing particular components of the loss function, such as the claim-fact distillation loss, intra-sample contrastive loss, and scoring loss. To do so, I set their respectively λ factors to 0, effectively nullifying their influence during training. I find that these loss components lead to significant performance decreases when removed, therefore justifying my architectural decisions.

4.5 Discussion and summary

In this chapter, I introduced a novel weakly supervised training method that leverages language model semantic distillation for dense retrieval. My approach is designed to overcome the limitations of annotation-dependent methods by automatically learning high-quality feature

representations for claim–fact matching without relying on human labels. The key innovation of my method lies in its ability to distill the rich semantic knowledge embedded in large pre-trained language models into a more compact, knowledge-aware model through novel contrastive and scoring losses.

My experiments on standard benchmarks such as FEVER and FB15k-237 demonstrate that the proposed framework not only achieves state-of-the-art performance but also substantially narrows the gap between supervised and unsupervised approaches. By distilling features from multiple language model backbones, I ensure that my method is both robust and versatile, capable of extracting meaningful representations even in low-data regimes. The significant improvements in label accuracy and retrieval metrics underscore the effectiveness of integrating claim-fact distillation, intra-sample contrastive learning, and scoring losses into a unified training objective.

The comprehensive ablation studies further validate my design choices, revealing that each loss component contributes critically to the overall performance. In particular, the contrastive mechanisms guide the model to distinguish between relevant and irrelevant evidence, while the scoring loss fine-tunes the ranking of evidence to ensure that positive evidence are prioritized. These insights highlight the importance of task-specific loss formulations that exploit the inherent inductive biases of evidence retrieval tasks.

Overall, my work presents a general and effective strategy for weakly supervised training. By enabling the transfer of semantic knowledge from large-scale language models into specialized knowledge models, my framework not only advances the state of the art in dense retrieval but also opens new avenues for future research in unsupervised and weakly supervised learning. This approach has the potential to be extended to other domains where obtaining labelled data is challenging, thereby fostering more scalable and efficient learning systems.

CHAPTER 5

TABULAR MANIFOLD DATA AUGMENTATION LEVERAGING PRE-TRAINED TRANSFORMERS

5.1 Introduction and contribution overview

Tabular data is prevalent and integral to critical domains such as medicine [207–209], physics [210, 211] and chemistry [212, 213]. However, acquiring large datasets can be prohibitively expensive or impossible, making it challenging to train machine learning models effectively [214, 215]. In vision and language tasks, a common remedy for addressing model performance issues due to data scarcity is employing data augmentation (DA) techniques to generate additional synthetic samples [216]. Techniques such as exploiting symmetries in the data, like rotating, flipping or altering the colour of images, or generating paraphrased sentences for text, are used to increase the diversity of the training set, and often lead to improved model generalisation. However, applying DA to tabular data in the input space is challenging, because tabular data is heterogeneous and lacks clear symmetries [217]. Consequently, existing tabular augmentation methods in the input space often degrade model performance, hindering their widespread adoption [218].

Data augmentation on a learned manifold is advantageous, because it enables label-invariant transformations [219, 220]. Manifolds provide a structured, continuous space where data points can be meaningfully interpolated or transformed. However, current manifold data augmentation (MDA) methods face two challenges in the context of tabular data. Firstly, these methods are typically model-specific, and no general MDA method is applicable across different models. Secondly, MDA has been primarily developed for neural networks, which learn a manifold space [221]. However, for tabular data, neural networks are often outperformed by more traditional methods such as tree-based algorithms like XGBoost [222] or logistic regression, which operate directly in the input space and do not learn a manifold during training. Therefore, it remains an

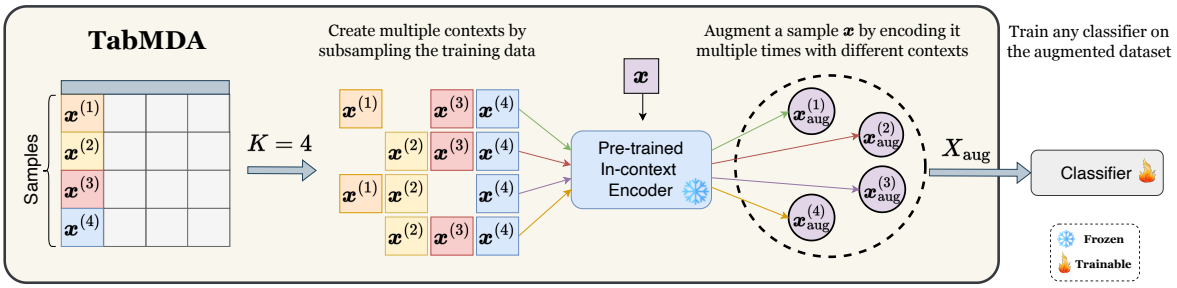

Figure 5.1: TabMDA improves tabular data classifiers by jointly embedding and augmenting the dataset in the embedding space of pre-trained tabular transformers using in-context learning. It generates multiple embeddings for each input by presenting different contexts to the encoder, leveraging its in-context learning capability. This results in an expanded training dataset with more diverse samples, enhancing the accuracy and robustness of the downstream predictor. TabMDA is training-free and can be applied to any classifier.

open question how to enhance these traditional methods with the benefits of MDA.

To address the challenges of small tabular datasets, I introduce TabMDA (Figure 5.1), a new training-free method for performing manifold data augmentation (MDA) on tabular data. TabMDA is a general method that works with any downstream classifier. It has two components: first, it leverages an existing pre-trained in-context model, such as TabPFN [223], to embed the data in a manifold space. Second, I introduce in-context subsetting (ICS), a novel technique that encodes the input data multiple times using the in-context encoder, with each iteration using different data subsets as context for the model. This process explores the manifold space learned by the pre-trained encoder (see Figure 5.2 for an illustration). The downstream classifier is then trained on this expanded manifold dataset, thereby indirectly incorporating the pre-training knowledge into any subsequent classification model. Crucially, my method requires no additional training and is applicable to any classifier. I investigate TabMDA using five common tabular classifiers across 28 different dataset configurations of varying sizes. My results show that TabMDA substantially improves the performance of various standard tabular classifiers, including tree-based ensembles such as XGBoost, while reducing the variability among their performance. As a result, TabMDA enables explainable classifiers, such as KNN, to become very competitive and sometimes even achieve the highest performance.

My contributions can be summarised as:

- **TabMDA:** A novel training-free data augmentation method that jointly embeds and augments the data using pre-trained in-context models. TabMDA can be applied to any classifier, and my results demonstrate that it improves performance across classifiers on small datasets.
- **In-context Subsetting:** A new technique for label-invariant transformations in the manifold of in-context models by using multiple iterations of in-context encoding with different data contexts.

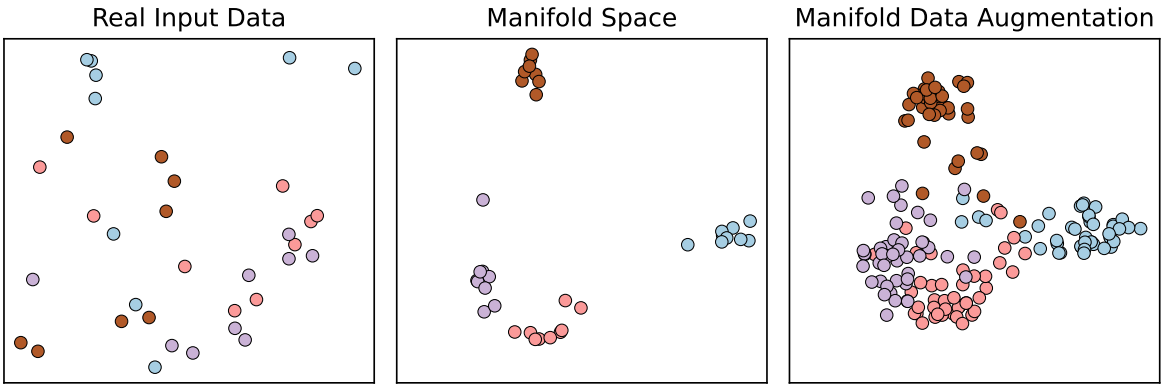

Figure 5.2: PCA linear projection of the first two PCs of the raw input data space for the "vehicle" dataset **(left)**, the manifold space using the encoder from [223] **(middle)**, and the augmented manifold space after using my proposed method **(right)**. The colour of the data points represents one of four class labels. Visualisations for all datasets are available in Section 5.5.4.

5.2 Related work

Using LLMs on tabular data. The remarkable success of Large Language Models (LLMs), trained on massive text corpora and scaled to unprecedented sizes [3, 224], has motivated their adaptation to tabular learning [225]. Adapting LLMs for tabular data leverages the broad world knowledge already learned, enables in-context learning (ICL) capabilities, and effectively utilises meta-information in tabular data, such as column names [226]. For instance, LIFT [227] introduced language-interfaced fine-tuning by adapting GPT-3 [3] and GPT-J [228] to multiple tabular learning datasets, finding that the performance of fine-tuned LLMs was roughly comparable to traditional solutions. TabLLM [229] fine-tuned T0 [230], showing slight underperformance compared to classical tree models. Although these studies demonstrate that while LLMs trained on text corpora can be applied to tabular learning, there remains a performance gap between LLMs and classical tabular models due to the lack of tabular-specific inductive biases in LLMs.

Transformer-based models for tabular data. Transformers for tabular data are typically developed by training from scratch on the target task. Occasionally, an additional pre-training phase on the target task itself is included, unlike LLMs, which use large external corpora. These models introduce various attention mechanisms to handle the tabular data structure. For instance, TUTA [231] introduces a novel attention mechanism that allows the model to attend to the entire table at once, while TAPAS [232] introduces a column-specific attention mechanism that allows the model to attend to the relevant columns in the table. TransTab [233] combines column descriptions and cells as input to a gated Transformer model for feature encoding. SAINT [234] performs attention across both rows and columns to capture the relationships between different cells in the table. More notable examples in this type of models include TabNet [235], TabTransformer [236], and FT-Transformer [237]. Even though transformers can be competitive on large tabular tasks but are often outperformed by well-tuned MLPs [238], especially on small

datasets [239]. This suggests that models trained on a single tabular task lack adequate in-context learning (ICL) capabilities for tabular data.

In-context learning (ICL) for tabular data and TabPFN. ICL allows Transformer-based models to adapt to new tasks by conditioning on demonstrations of input-output pairs. Recently, ICL has shown promising results on tabular data. Notably, TabPFN [223], a meta-learned Transformer with ICL capabilities, has demonstrated competitive accuracy compared to tree-based ensembles on small supervised classification tasks. TabPFN employs a Prior-Data Fitted Network (PFN) [240] approach, where it is pre-trained on synthetic datasets generated from Structural Causal Models (SCMs) and Bayesian Neural Networks (BNNs), which resembles the causal structure commonly assumed in tabular data. This pretraining enables TabPFN to approximate Bayesian inference in a single forward pass, reducing computational needs and eliminating hyper-parameter tuning. During inference, TabPFN uses the training dataset (context) and new sample features as input, directly outputting the posterior predictive distribution (PPD) for the new samples. This leverages in-context learning to embed new samples without parameter updates, making TabPFN effective on small datasets. Although TabPFN can embed samples into a rich manifold space using its in-context learning abilities, the nature of this embedding space has not been studied. However, I believe these manifolds could facilitate data augmentation for various classifiers.

Tabular Data Augmentation. Tabular data augmentation is less explored due to several challenges, such as the lack of explicit symmetries like rotations or translations and generally weaker feature correlations compared to the strong spatial or semantic relationships in image or text data [241]. Augmenting tabular data often involves training generative models, which is computationally expensive [242–246]. On small tasks, this can lead to overfitting [247] and mode collapse [248, 249], where the model fails to capture the diversity of each class. Recently, TabPFGen [250] transformed TabPFN into an energy-based class-conditional generator for data augmentation in the input-space. While it does not require specialised training, TabPFGen achieves minimal performance improvements and retains TabPFN's limitations, such as being applicable to only up to ten classes. In contrast, my method, TabMDA, is capable of handling datasets with an unlimited number of classes. Overall, existing methods for augmenting tabular data often yield mixed results and can even degrade model performance [218, 251].

5.3 Preliminaries

5.3.1 Problem formulation

I focus on supervised classification tasks involving $\mathcal{Y}$ categorical classes. Formally, let the training set be defined as $\{(\boldsymbol{x}^{(i)}, y_i)\}_{i=1}^{N}$, composed of N samples, where each sample $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$ is continuous and associated with a categorical label $y_i \in \mathcal{Y}$. For convenience, I represent the

training dataset as a feature matrix $\boldsymbol{X}_{\text{train}} \in \mathbb{R}^{N \times D}$ and a corresponding label vector $\boldsymbol{y}_{\text{train}} := [y_1, y_2, ..., y_N]$.

5.3.2 Pretrained Transformers for tabular data

Transformer architectures [1] have demonstrated remarkable success in diverse domains such as language and vision. Recently, Transformers have also been successfully adapted to handle tabular data [233, 234, 236]. Unlike typical Transformers, pretrained tabular Transformers are specifically designed to capture the inherent relationships and interactions between features within structured tabular datasets. These models are generally pretrained using extensive synthetic or real-world tabular datasets, endowing them with latent representations of tabular structures beneficial for downstream tasks.

A notable example of such pretrained tabular Transformers is the Tabular Prior-Data Fitted Network (TabPFN) [223], which is employed in this work. TabPFN is pretrained on synthetic datasets generated from Structural Causal Models (SCMs) and Bayesian Neural Networks (BNNs), thereby effectively capturing causal structures commonly found in tabular data. TabPFN approximates Bayesian inference in a single forward pass, significantly reducing computational demands and eliminating the need for hyper-parameter tuning. At inference time, TabPFN embeds input samples in a latent manifold space using an attention-based mechanism, conditioned on provided input samples and their corresponding labels (context). This unique capability, known as in-context learning (ICL), makes TabPFN particularly effective for small-to-medium-sized tabular datasets, providing rich, context-dependent latent representations.

5.4 Methodology

In this section, I describe TabMDA (Figure 5.1), a novel training-free manifold data augmentation method for tabular data. TabMDA enhances downstream performance of any classifier by augmenting the dataset through diverse label-invariant transformations in the embedding space of pretrained tabular Transformers, leveraging the context-dependent nature of embeddings produced by these models.

5.4.1 Embedding tabular data into the manifold space

TabMDA begins by embedding the real data into a manifold space of a pre-trained Transformer-based tabular in-context model. Specifically, I use the pretrained TabPFN encoder [223], as discussed in the preliminaries, to encode input data points into a latent embedding space.

In particular, I utilise only the encoder component of the TabPFN model for embedding data points $\boldsymbol{x}$ in its latent space of size D'. The encoder takes the input samples $\boldsymbol{X}_{\text{train}}$ and their labels y_{train} as the context. For clarity, I denote the context $\boldsymbol{X}_{\text{ctx}}$ using only the input samples

Algorithm 1 In-context Subsetting (ICS)

Input: Tabular in-context model ϕ; Number of contexts to generate K; Context size N_{ctx}; Train data $\boldsymbol{X}_{\text{train}}$: Data point to augment $\boldsymbol{x}$ **Output:** Augmented versions of $\boldsymbol{x}$

function ICS($\phi, K, N_{\text{ctx}}, \boldsymbol{X}_{\text{train}}, \boldsymbol{x}$)
  for $k = 1$ to K **do**
    $\boldsymbol{X}_{\text{ctx}}^{(k)}$ = Stratified sample N_{ctx} points from $\boldsymbol{X}_{\text{train}}$
    $\boldsymbol{x}_{\text{aug}}^{(k)} = \phi(\boldsymbol{X}_{\text{ctx}}^{(k)}, \boldsymbol{x})$ ▷ Augment the data point
  end for
  Let $\boldsymbol{X}_{\text{aug}} := [\boldsymbol{x}_{\text{aug}}^{(1)}, \boldsymbol{x}_{\text{aug}}^{(2)}, \ldots, \boldsymbol{x}_{\text{aug}}^{(K)}]$
  return $\boldsymbol{X}_{\text{aug}}$ as the augmented versions of $\boldsymbol{x}$
end function

$\boldsymbol{X}_{\text{train}}$. Given a set of data consisting of samples $\boldsymbol{x}$, the encoder produces embeddings $\phi(\boldsymbol{X}_{\text{ctx}}, \boldsymbol{x})$ that depend on the context provided during encoding. This context-dependent encoding arises naturally from the model self-attention mechanism, which computes representations of data points conditioned on their provided context. Consequently, using the model to encode with different contexts will produce different embeddings for the same data point $\boldsymbol{x}$.

5.4.2 Manifold data augmentation with in-context subsetting

I propose a novel data augmentation technique named in-context subsetting (ICS), which leverages the contextual sensitivity of pretrained Transformer encoders to perform label-invariant transformations directly within the latent manifold space. Given a training dataset $\boldsymbol{X}_{\text{train}}$ and a sample $\boldsymbol{x}$ to augment, ICS constructs multiple contexts by randomly selecting subsets from $\boldsymbol{X}_{\text{train}}$ and encoding the target data point within these varying contexts.

Formally, for each data point $\boldsymbol{x}$, I sample K distinct contexts $\boldsymbol{X}_{\text{ctx}}^{(k)}$, each consisting of N_{ctx} data points sampled via stratified sampling from the training dataset. Encoding $\boldsymbol{x}$ using each context $\boldsymbol{X}_{\text{ctx}}^{(k)}$ results in K augmented embeddings, denoted as $\boldsymbol{X}_{\text{aug}} := [\boldsymbol{x}_{\text{aug}}^{(1)}, \boldsymbol{x}_{\text{aug}}^{(2)}, \ldots, \boldsymbol{x}_{\text{aug}}^{(K)}]$. This procedure enriches the training dataset with diverse manifold embeddings, enhancing the downstream model capacity to generalize. Figure 5.2 provides an illustration of the effect of manifold data augmentation. Algorithm 1 provides a description of the ICS algorithm.

5.4.3 Training downstream classifiers on augmented data

With TabMDA, I construct an augmented training dataset by embedding and augmenting each data point using ICS. Specifically, I encode each training point $\boldsymbol{x}^{(i)}$ to obtain the augmented training dataset:

$$\boldsymbol{X}_{\text{train}}^{\text{augm}} := \left\{ \left(\text{ICS}\left(\phi, K, N_{\text{ctx}}, \boldsymbol{X}_{\text{train}}, \boldsymbol{x}^{(i)}\right), \underbrace{y_i, \ldots, y_i}_{K \text{ times}} \right) \right\}_{i=1}^{N} \tag{5.1}$$

where $\boldsymbol{X}_{\text{train}}^{\text{augm}} \in \mathbb{R}^{(K \times N) \times D'}$ represents the expanded, diverse embedding dataset and D'

is the dimensionality of the embedding manifold. Due to the label-invariant nature of ICS, augmented samples retain their original labels. The downstream classifier, whether neural or classical (e.g., XGBoost, KNN), is subsequently trained exclusively on this augmented dataset. At test time, points $\boldsymbol{x}^{\text{test}}$ are embedded using the complete training context (equivalent to $\text{ICS}(\phi, 1, N, \boldsymbol{X}_{\text{train}}, \boldsymbol{x}^{\text{test}})$), ensuring consistent manifold alignment between train and test embeddings. As demonstrated in the experiments in the next section, TabMDA consistently improves generalisation and robustness across various downstream classifiers, reducing performance variability, and notably enhancing simpler, interpretable models.

5.5 Experiments

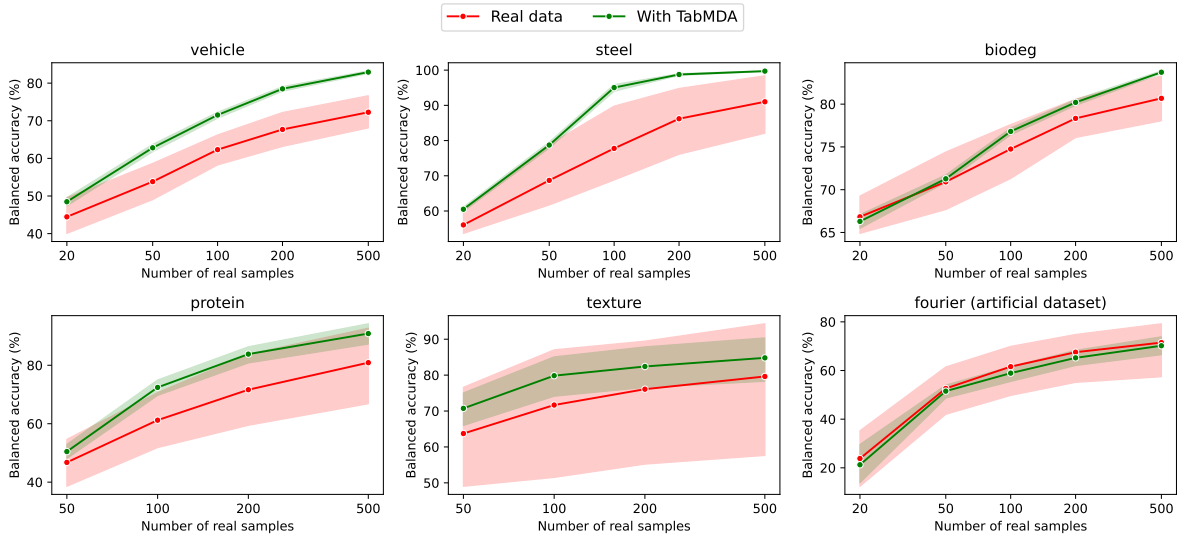

Figure 5.3: Average accuracy (%) for five downstream classifiers trained on real data (the original input space) and TabMDA embeddings. I report the mean±std of test balanced accuracy over 10 runs for each predictor, totalling 50 runs. Training with TabMDA substantially improves performance across five real-world tabular datasets and reduces variability among classifiers. It also performs well on artificial datasets, such as "fourier", which are processed images, despite differing from the inductive bias of the in-context encoder, TabPFN.

5.5.1 Experimental setup

Datasets. I focus on small-to-medium classification tasks, because this regime allows the study of the effect of data augmentation. I consider six tabular datasets, publicly available on OpenML [252] with complete details included in Table 5.1. I evaluate all models over 10 runs and report the mean and standard deviation of the test-balanced accuracy averaged across them.

Dataset	OpenML ID	# samples (N)	# features (D)	N/D	# classes	# min samples per class	# max samples per class
vehicle	54	846	18	47	4	199	218
steel	1504	1941	33	59	2	673	1268
biodeg	1494	1055	41	26	2	356	699
protein	40966	1080	77	14	8	105	150
texture	40499	5500	40	138	11	500	500
fourier	14	2000	76	26	10	200	200

Table 5.1: Details of the datasets used for experiments. "fourier" is an artificial dataset created by processing images.

Setup and evaluation. For each dataset comprising N samples, I initially split it into stratified training and test sets. I use a sizeable test set to ensure an accurate evaluation of the model's performance. The size of the test set is calculated as $N_{\text{test}} = \min\left(\frac{N}{2}, 500\right)$. To better understand the impact of my method across different dataset sizes, I subsample the full training set to simulate varying levels of data availability, creating subsets with sample sizes $N_{\text{real}} \in \{20, 50, 100, 200, 500\}$. Each subset is split into training and validation sets, with 80% of N_{real} for training and the remaining 20% for validation. I repeat the training set splitting 10 times to result in 10 runs per subset size. Note that the same test set is used for all repeats to ensure consistency in performance evaluation.

Implementation of classifier models. For implementing the various classifier models in my experiments, I utilised established libraries to ensure reliability and consistency. The K-Nearest Neighbors (KNN), Logistic Regression, Decision Tree, and Random Forest classifiers were all implemented using the scikit-learn library [253] (BSD license), version 1.3.2. For the XGBoost classifier, I used the xgboost library, version 1.7.6, which is specifically optimised for gradient boosting.

Hyper-parameters for benchmark models. The K-Nearest Neighbors (KNN) classifier was configured with 5 neighbors and used the cosine distance metric for measuring similarity between instances. For Logistic Regression, I set the maximum number of iterations to 1000 to allow sufficient convergence during optimisation. The Decision Tree classifier was limited to a maximum depth of 3, with a minimum of 2 samples required per leaf to prevent overfitting. Similarly, the Random Forest classifier comprised 200 trees, each with a maximum depth of 3 and a minimum of 2 samples per leaf, balancing complexity and generalisation. Lastly, the XGBoost classifier was configured with 200 estimators, a learning rate of 0.3, and a maximum depth of 3.

TabMDA settings. For the in-context model, I use the encoder component of TabPFN [223], a pre-trained classifier. I utilise the embeddings produced by its Transformer encoder and I use the official weights released by the TabPFN authors. I use TabPFN without ensembling (referred to as TabPFN$_{n.e.}$ in the original paper), and I do not disable its internal data preprocessing. For the proposed in-context subsetting, I define the context size as a proportion of the training set, considering context sizes of 0.5, 0.7, 0.9, and 1. Additionally, I evaluate 5, 20, and 50 subcontexts.

Dataset	N_{real}	TabPFN	KNN		Logistic Regression		Decision Tree		Random Forest		XGBoost	
			Real data	TabMDA	Real data	TabMDA	Real data	TabMDA	Real data	TabMDA	Real data	TabMDA
vehicle	20	$50.81_{\pm 6.38}$	$38.68_{\pm 3.59}$	$51.12_{\pm 4.44}$ ↑	$52.40_{\pm 3.64}$	$51.69_{\pm 3.44}$	$38.87_{\pm 5.91}$	$41.44_{\pm 6.76}$ ↑	$47.16_{\pm 3.95}$	**$52.84_{\pm 5.28}$** ↑	$45.21_{\pm 4.60}$	$51.54_{\pm 5.44}$ ↑
	50	$64.55_{\pm 5.54}$	$49.32_{\pm 3.51}$	$64.93_{\pm 3.66}$ ↑	$60.99_{\pm 3.87}$	**$65.62_{\pm 3.52}$** ↑	$46.42_{\pm 5.40}$	$59.89_{\pm 4.93}$ ↑	$55.76_{\pm 3.60}$	$64.21_{\pm 3.55}$ ↑	$56.60_{\pm 4.03}$	$63.64_{\pm 4.73}$ ↑
	100	**$73.67_{\pm 2.93}$**	$59.04_{\pm 3.86}$	$73.31_{\pm 1.94}$ ↑	$68.80_{\pm 2.80}$	$72.10_{\pm 2.97}$ ↑	$55.52_{\pm 6.49}$	$69.76_{\pm 1.88}$ ↑	$62.56_{\pm 5.54}$	$73.48_{\pm 3.20}$ ↑	$65.65_{\pm 3.67}$	$71.66_{\pm 2.24}$ ↑
	200	$81.13_{\pm 1.87}$	$64.31_{\pm 2.78}$	$77.74_{\pm 1.75}$ ↑	$74.26_{\pm 1.82}$	$80.77_{\pm 1.95}$ ↑	$60.15_{\pm 3.34}$	$76.72_{\pm 2.29}$ ↑	$66.96_{\pm 3.57}$	**$81.58_{\pm 2.20}$** ↑	$72.74_{\pm 2.00}$	$80.50_{\pm 2.56}$ ↑
	500	$84.11_{\pm 1.00}$	$68.90_{\pm 0.92}$	$83.33_{\pm 1.02}$ ↑	$78.29_{\pm 1.11}$	$84.57_{\pm 0.85}$ ↑	$65.75_{\pm 2.85}$	$81.14_{\pm 1.16}$ ↑	$70.07_{\pm 0.94}$	**$85.11_{\pm 1.32}$** ↑	$78.30_{\pm 1.67}$	$84.12_{\pm 1.21}$ ↑
steel	20	$57.96_{\pm 4.29}$	$54.17_{\pm 3.72}$	$61.62_{\pm 3.62}$ ↑	$63.96_{\pm 9.21}$	**$65.10_{\pm 4.00}$** ↑	$53.92_{\pm 3.12}$	$61.08_{\pm 6.68}$ ↑	$53.23_{\pm 2.73}$	$64.76_{\pm 5.27}$ ↑	$54.94_{\pm 3.48}$	$62.03_{\pm 4.41}$ ↑
	50	$82.09_{\pm 7.91}$	$69.81_{\pm 5.54}$	$81.75_{\pm 7.27}$ ↑	**$88.06_{\pm 5.64}$**	$82.42_{\pm 9.58}$	$62.68_{\pm 9.97}$	$76.10_{\pm 7.92}$ ↑	$60.40_{\pm 3.43}$	$82.69_{\pm 8.38}$ ↑	$62.53_{\pm 2.57}$	$81.55_{\pm 7.17}$ ↑
	100	$97.37_{\pm 1.37}$	$81.55_{\pm 4.73}$	$97.76_{\pm 1.28}$ ↑	**$98.85_{\pm 1.20}$**	$97.78_{\pm 1.46}$	$70.78_{\pm 12.39}$	$95.25_{\pm 2.98}$ ↑	$63.89_{\pm 2.75}$	$98.04_{\pm 1.29}$ ↑	$73.79_{\pm 6.19}$	$96.99_{\pm 1.27}$ ↑
	200	$98.84_{\pm 0.70}$	$87.58_{\pm 2.98}$	**$99.57_{\pm 0.62}$** ↑	$99.43_{\pm 0.58}$	$99.55_{\pm 0.61}$ ↑	$84.40_{\pm 8.69}$	$97.83_{\pm 1.86}$ ↑	$68.29_{\pm 2.91}$	$99.31_{\pm 0.69}$ ↑	$91.27_{\pm 3.47}$	$98.79_{\pm 1.01}$ ↑
	500	$99.77_{\pm 0.30}$	$93.51_{\pm 1.22}$	**$99.91_{\pm 0.14}$** ↑	$99.78_{\pm 0.27}$	$99.83_{\pm 0.24}$ ↑	$88.01_{\pm 0.71}$	$99.77_{\pm 0.23}$ ↑	$74.22_{\pm 2.02}$	$99.83_{\pm 0.24}$ ↑	$99.55_{\pm 0.72}$	$99.67_{\pm 0.50}$ ↑
biodeg	20	$66.85_{\pm 9.33}$	$67.24_{\pm 6.19}$	$69.64_{\pm 5.19}$ ↑	$71.63_{\pm 5.43}$	$69.04_{\pm 6.99}$	$64.15_{\pm 7.02}$	$63.48_{\pm 7.03}$	$65.03_{\pm 7.75}$	**$72.06_{\pm 4.04}$** ↑	$66.10_{\pm 6.95}$	$69.82_{\pm 4.61}$ ↑
	50	$75.21_{\pm 2.67}$	$73.27_{\pm 2.31}$	$72.37_{\pm 3.37}$	**$76.42_{\pm 3.03}$**	$74.83_{\pm 3.24}$	$64.85_{\pm 3.55}$	$70.01_{\pm 6.57}$ ↑	$69.47_{\pm 3.52}$	$73.28_{\pm 6.31}$ ↑	$70.56_{\pm 4.05}$	$72.90_{\pm 6.44}$ ↑
	100	$78.78_{\pm 1.56}$	$76.98_{\pm 1.77}$	$78.59_{\pm 1.92}$ ↑	**$79.06_{\pm 1.66}$**	$77.37_{\pm 2.61}$	$68.86_{\pm 3.61}$	$77.08_{\pm 3.57}$ ↑	$73.88_{\pm 2.36}$	$77.85_{\pm 2.77}$ ↑	$74.98_{\pm 2.08}$	$78.02_{\pm 2.84}$ ↑
	200	**$82.66_{\pm 1.87}$**	$79.86_{\pm 2.50}$	$82.64_{\pm 1.80}$ ↑	$81.92_{\pm 1.56}$	$80.68_{\pm 2.44}$	$75.12_{\pm 3.24}$	$80.17_{\pm 4.61}$ ↑	$75.48_{\pm 2.09}$	$81.34_{\pm 1.04}$ ↑	$79.30_{\pm 1.41}$	$80.13_{\pm 1.84}$ ↑
	500	$84.96_{\pm 0.67}$	$82.48_{\pm 0.65}$	**$85.05_{\pm 0.81}$** ↑	$83.86_{\pm 0.54}$	$84.47_{\pm 1.24}$ ↑	$77.12_{\pm 1.89}$	$82.54_{\pm 1.13}$ ↑	$76.78_{\pm 2.01}$	$84.45_{\pm 0.68}$ ↑	$83.17_{\pm 1.63}$	$84.09_{\pm 1.01}$ ↑
protein	50	$51.65_{\pm 5.74}$	$36.68_{\pm 3.40}$	$55.82_{\pm 4.60}$ ↑	$61.22_{\pm 3.56}$	**$62.48_{\pm 4.21}$** ↑	$38.12_{\pm 2.82}$	$40.51_{\pm 5.98}$ ↑	$52.98_{\pm 3.13}$	$54.90_{\pm 4.22}$ ↑	$44.78_{\pm 4.01}$	$56.20_{\pm 5.22}$ ↑
	100	$72.58_{\pm 3.76}$	$50.28_{\pm 3.67}$	$79.26_{\pm 2.66}$ ↑	$79.46_{\pm 3.22}$	**$80.44_{\pm 2.67}$** ↑	$48.11_{\pm 5.03}$	$57.89_{\pm 3.25}$ ↑	$63.69_{\pm 2.74}$	$77.32_{\pm 2.96}$ ↑	$64.52_{\pm 3.21}$	$79.40_{\pm 2.73}$ ↑
	200	$86.69_{\pm 2.77}$	$66.42_{\pm 2.62}$	$91.05_{\pm 1.81}$ ↑	$90.98_{\pm 1.79}$	**$92.97_{\pm 1.91}$** ↑	$50.50_{\pm 2.26}$	$65.40_{\pm 4.73}$ ↑	$70.89_{\pm 3.18}$	$88.40_{\pm 1.41}$ ↑	$79.38_{\pm 1.44}$	$91.49_{\pm 1.34}$ ↑
	500	$95.58_{\pm 0.98}$	$84.94_{\pm 1.58}$	$98.09_{\pm 0.53}$ ↑	$97.85_{\pm 0.81}$	**$98.21_{\pm 0.59}$** ↑	$53.48_{\pm 3.19}$	$70.43_{\pm 1.54}$ ↑	$75.90_{\pm 2.09}$	$97.27_{\pm 0.93}$ ↑	$92.29_{\pm 1.42}$	$98.06_{\pm 0.60}$ ↑
texture	50	N/A	$62.81_{\pm 3.09}$	$78.50_{\pm 3.85}$ ↑	$86.07_{\pm 2.67}$	**$88.08_{\pm 2.53}$** ↑	$37.34_{\pm 7.19}$	$44.49_{\pm 4.58}$ ↑	$70.41_{\pm 2.01}$	$79.24_{\pm 3.31}$ ↑	$62.03_{\pm 4.78}$	$83.48_{\pm 3.25}$ ↑
	100	N/A	$77.84_{\pm 2.29}$	$93.75_{\pm 1.48}$ ↑	$94.05_{\pm 1.75}$	**$95.52_{\pm 1.18}$** ↑	$34.84_{\pm 3.71}$	$35.55_{\pm 5.36}$ ↑	$76.42_{\pm 1.46}$	$90.67_{\pm 1.90}$ ↑	$75.22_{\pm 3.83}$	$94.04_{\pm 1.37}$ ↑
	200	N/A	$85.09_{\pm 1.69}$	$95.62_{\pm 1.40}$ ↑	$96.55_{\pm 1.18}$	**$97.49_{\pm 0.85}$** ↑	$38.12_{\pm 4.61}$	$39.73_{\pm 3.88}$ ↑	$76.65_{\pm 1.34}$	$94.25_{\pm 1.22}$ ↑	$84.00_{\pm 1.83}$	$96.39_{\pm 1.19}$ ↑
	500	N/A	$91.28_{\pm 1.32}$	$98.11_{\pm 0.25}$ ↑	$98.01_{\pm 0.35}$	**$98.73_{\pm 0.47}$** ↑	$39.67_{\pm 3.72}$	$41.46_{\pm 7.53}$ ↑	$77.34_{\pm 2.20}$	$97.40_{\pm 0.63}$ ↑	$91.83_{\pm 1.29}$	$98.45_{\pm 0.51}$ ↑
fourier (artificial)	20	$26.58_{\pm 5.16}$	$19.58_{\pm 2.17}$	$18.32_{\pm 3.04}$	**$42.88_{\pm 5.23}$**	$35.00_{\pm 5.65}$	$11.26_{\pm 2.78}$	$13.72_{\pm 2.67}$ ↑	$35.42_{\pm 3.19}$	$29.29_{\pm 3.98}$	$10.00_{\pm 0.00}$	$10.00_{\pm 0.00}$
	50	$55.38_{\pm 4.83}$	$54.38_{\pm 2.56}$	$57.16_{\pm 3.99}$ ↑	$60.66_{\pm 1.64}$	$63.38_{\pm 2.51}$ ↑	$31.64_{\pm 4.53}$	$36.64_{\pm 3.62}$ ↑	**$65.78_{\pm 3.55}$**	$55.16_{\pm 4.32}$	$50.62_{\pm 5.22}$	$53.72_{\pm 3.01}$ ↑
	100	$61.98_{\pm 3.40}$	$63.24_{\pm 2.17}$	$68.54_{\pm 2.45}$ ↑	$67.84_{\pm 2.50}$	$70.10_{\pm 3.07}$ ↑	$38.68_{\pm 6.94}$	$40.28_{\pm 4.41}$ ↑	**$73.20_{\pm 2.61}$**	$60.48_{\pm 3.24}$	$64.96_{\pm 3.16}$	$67.86_{\pm 3.16}$ ↑
	200	$68.94_{\pm 2.41}$	$70.74_{\pm 2.57}$	$75.10_{\pm 2.06}$ ↑	$72.84_{\pm 2.43}$	$74.62_{\pm 2.36}$ ↑	$43.62_{\pm 5.61}$	$42.94_{\pm 5.87}$	**$76.42_{\pm 2.53}$**	$68.24_{\pm 1.14}$	$74.00_{\pm 1.56}$	$74.74_{\pm 1.67}$ ↑
	500	$76.28_{\pm 2.07}$	$78.08_{\pm 1.04}$	$81.22_{\pm 0.93}$ ↑	$77.52_{\pm 1.17}$	$79.88_{\pm 1.76}$ ↑	$43.82_{\pm 6.07}$	$41.98_{\pm 6.59}$	$77.64_{\pm 1.51}$	$76.42_{\pm 1.86}$	$80.30_{\pm 1.51}$	**$81.24_{\pm 0.76}$** ↑
Average accuracy		73.77	67.43	77.50↑	78.70	79.38↑	53.06	60.83↑	67.00	77.14↑	69.59	77.16↑

Table 5.2: Classification accuracy (%) of five downstream predictors trained on real data (the original input space) and on TabMDA embeddings. I present the mean ± standard deviation of the test balanced accuracy averaged over ten runs. ↑ indicates that TabMDA improves the performance of the downstream classifier. N/A indicates that TabPFN is not applicable to "texture". I **bold** the highest accuracy for each dataset of different sample sizes. My method, TabMDA, consistently improves the accuracy of downstream classifiers, with an average improvement of up to 10%.

I select the optimal TabMDA settings based on the validation balanced accuracy for each dataset configuration.

5.5.2 Overall performance on classification accuracy

I assess the impact of TabMDA on the performance of various tabular classifiers. I train these classifiers on real data (the original input space) and on TabMDA-augmented data, which consists of augmented embeddings output from the TabPFN's Transformer encoder. I consider five classification models with different inductive biases: distance-based methods (K-Nearest Neighbors (KNN)), linear methods (Logistic Regression) and tree-based methods (Decision Tree [254], Random Forest [255] and XGBoost [222]). I could not use TabPFN for the final prediction because it handles data with up to 100 dimensions, and the embeddings provided by TabMDA have 512 dimensions. Following previous work [256], I train the predictors with a fixed set of hyper-parameters. I use identical hyper-parameters for the downstream models when trained on real data and with TabMDA.

On the overall downstream performance. Figure 5.3 illustrates the overall impact of training downstream predictors with TabMDA. The results indicate that TabMDA enhances

overall performance by up to 15% across five real-world tabular datasets. Improvements are observed across all dataset sizes, with more significant gains for datasets of 100 samples or more. On average, 3% of this performance improvement is attributed to the proposed in-context subsetting (regardless of the context sizes), and the remaining performance gain is due to training directly on the TabPFN's manifold (see Section 5.5.3). Higher context sizes of 0.7 and 0.9 tend to yield better results than the smaller context size of 0.5. This trend is particularly noticeable for small datasets, where a smaller context size might introduce too much variance, negatively impacting performance (see Section 5.5.3).

One important observation is that training with TabMDA significantly reduces performance variability across classifiers. For example, on the "vehicle", "steel" and "biodeg" datasets, classifiers trained on real data exhibit considerable performance variation, as indicated by the large standard deviations in accuracy (marked with red hue band in Figure 5.3). In contrast, training with TabMDA substantially reduces the performance variability, resulting in predictors with similar performance levels. This is advantageous as it enables using interpretable classifiers, such as KNN, which typically perform poorly when applied directly to the input space.

As expected, the performance of TabMDA depends on the underlying in-context model and its pre-training inductive bias (i.e., the model's prior). I use TabPFN, which was trained on data with a suitable prior for tabular data, allowing it to perform well with real-world datasets. Note that, from the six selected datasets in Figure 5.3, "fourier" is an artificial dataset consisting of Fourier coefficients from digitised images of handwritten numerals on Dutch maps. This dataset does not align well with TabPFN's prior, leading to minimal performance improvements by TabMDA. Despite this, TabMDA significantly enhances performance stability.

TabMDA improves performance across classifiers. The results in Table 5.2 show substantial improvements in classification accuracy across multiple datasets when using TabMDA embeddings compared to training on real data alone. TabMDA consistently enhances performance for common tree-based ensembles like Random Forest and XGBoost. Logistic Regression also sees improvements, albeit to a lesser extent. Notably, training with TabMDA achieves the highest performance in 18 out of 24 real-world dataset settings (as "fourier" is an artificial dataset).

KNN trained with TabMDA is surprisingly competitive. Simple classifiers like KNN are advantageous due to their explainability and ease of debugging. KNN makes predictions based on available training data, making them easy to understand and debug. These classifiers offer highly desirable properties, such as providing prototypical explanations of predictions. However, when trained on real data, KNN shows a notable performance gap of $\approx$9% when compared to other classifiers. The results in Table 5.2 show that KNN trained with TabMDA is surprisingly competitive, closely matching maximal performance. On a larger dataset of 500 samples, KNN (with TabMDA) achieves the best accuracy on two datasets and encounters less than 2% performance degradation on the other three. This result indicates that the manifolds

learned by the pre-trained in-context transforms possess meaningful distances, making it an interesting direction for future work.

5.5.3 Effect of in-context subsetting on classification performance

Table 5.3 shows the impact of in-context subsetting (ICS) on classification performance. I compared the performance of using the full context (i.e., no ICS) against ICS with various context sizes N_{ctx}. Even though the optimal context subset size varies across datasets, the results show ICS improves performance across datasets compared to using the full context. Higher context sizes of 0.7 and 0.9 tend to yield better results than the smaller context size of 0.5. This trend is particularly noticeable for small datasets, where a smaller context size might introduce too much variance, negatively impacting performance. Therefore, it is advisable to consider larger context sizes of 0.7 and 0.9 for improved classification accuracy, especially when dealing with limited data.

I recognize that one limitation of TabMDA is the need to tune the context size hyper-parameter. Inspired by TrivialAugment [257], I implemented a similar augmentation strategy within TabMDA. Instead of using a fixed context size, I randomly sample $N_{\text{ctx}} \sim U[0.5, 0.99]$ for each context subsetting. This approach (Column TabMDA (TA) in Table 5.3) consistently outperforms using the full context, but performs slightly worse than using a fixed context size value. Further investigation is necessary to potentially develop TabMDA into a hyper-parameter-free method.

Dataset	N_{real}	Full context (no ICS)	TabMDA (with ICS)			
			$N_{\text{ctx}} = 0.5$	$N_{\text{ctx}} = 0.7$	$N_{\text{ctx}} = 0.9$	$N_{\text{ctx}} \sim U[0.5, 0.99]$
vehicle	20	46.47	48.56	49.97	**50.43**	46.28
	50	60.03	63.39	**64.22**	63.52	61.61
	100	68.39	72.05	**72.71**	72.29	70.31
	200	77.14	78.7	**79.04**	78.85	78.03
	500	83.04	**83.07**	**83.07**	82.97	82.54
steel	20	60.04	59.62	59.4	**62.63**	60.73
	50	77.57	76.23	79.92	**80.9**	78.31
	100	91.57	94.9	96.6	**96.85**	94.32
	200	97.78	97.87	**99.22**	99.21	98.94
	500	99.27	99.85	**99.87**	99.73	99.68
biodeg	20	66.25	65.94	64.32	**69.55**	66.93
	50	**72.16**	69.74	71.53	71.92	71.15
	100	77.03	76.47	77.13	**77.33**	75.5
	200	**81.23**	80.68	80.86	80.28	79.05
	500	**84.15**	83.91	83.81	83.67	83.37
protein	50	45.55	50.96	53.2	**53.97**	48.83
	100	64.34	74.65	**74.75**	73.44	71.78
	200	77.9	85.24	**85.73**	82.97	83.99
	500	86.55	92.3	**92.41**	88.89	91.91
texture	50	66.0	**73.28**	72.1	72.1	69.04
	100	75.53	80.82	**81.07**	80.41	79.88
	200	80.66	82.55	**83.37**	82.93	82.03
	500	**85.78**	84.64	84.77	85.06	84.59
fourier (artificial)	50	44.02	51.04	**53.72**	53.69	50.94
	100	52.6	**60.76**	60.51	59.47	58.28
	200	61.05	**66.59**	66.44	65.35	64.73
	500	68.82	**70.63**	70.35	70.18	69.87
Average accuracy		72.26	74.98	75.56	75.50	74.17

Table 5.3: Ablation of the context size hyper-parameter for In-context Subsetting (ICS). The last column depicts a TrivialAugment-inspired version of TabMDA, where the context size is uniformly sampled from $N_{\text{ctx}} \sim U[0.5, 0.99]$ instead of fixed to a predefined value. ICS improves performance across datasets compared to using the full context.

5.5.4 Qualitative analysis of original data versus generated manifold space

To qualitatively illustrate the effectiveness of TabMDA, I visually compare the distribution of the original tabular data in the raw input space, the manifold embeddings generated by the pretrained TabPFN encoder, and the augmented manifold embeddings produced by my proposed TabMDA method. Figures 5.4, 5.5, 5.6, 5.7, and 5.8 clearly demonstrate that the augmented manifold spaces exhibit richer and more structured class separations compared to the raw input spaces. These visualisations support my quantitative results by highlighting how TabMDA generates embeddings that enhance class separability and diversity, facilitating improved downstream classification performance.

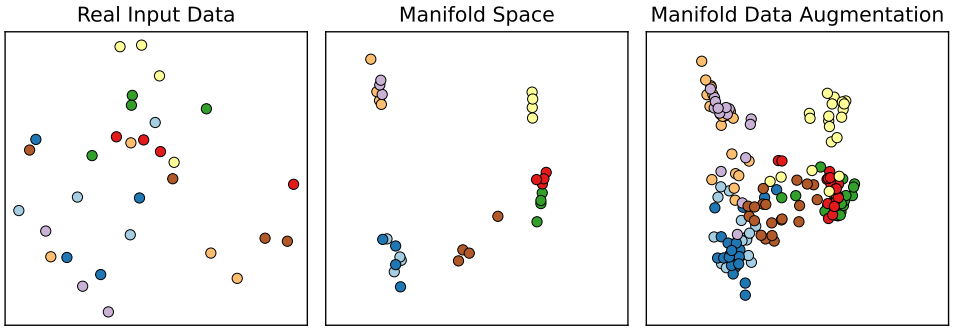

Figure 5.4: PCA linear projection of the first 2 PCs of the raw input data space for the "protein" dataset **(left)**, the manifold space using the encoder from [223] **(middle)** and the augmented manifold space after using my proposed method **(right)**. The colour of the data points depicts class label.

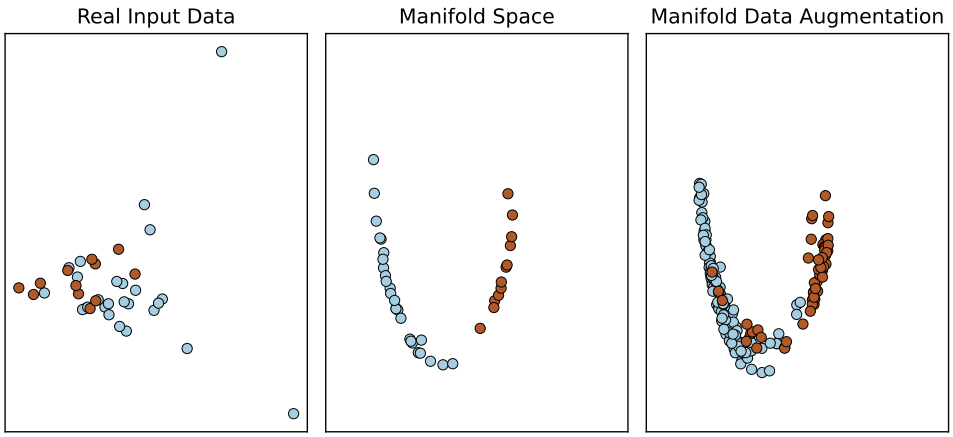

Figure 5.5: PCA linear projection of the first 2 PCs of the raw input data space for the "biodeg" dataset **(left)**, the manifold space using the encoder from [223] **(middle)** and the augmented manifold space after using my proposed method **(right)**. The colour of the data points depicts class label.

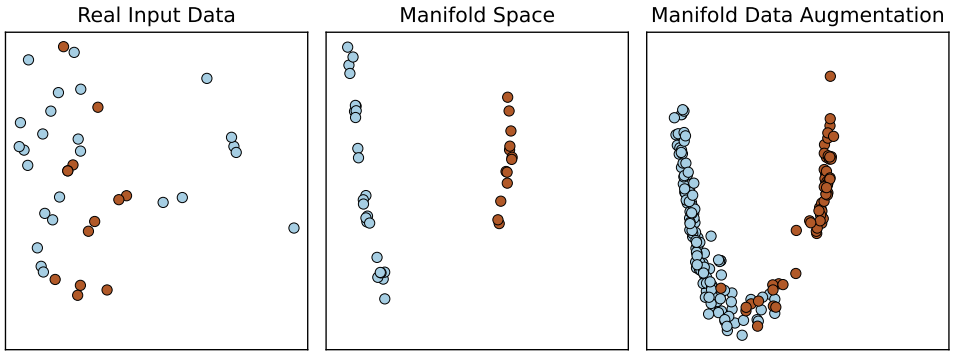

Figure 5.6: PCA linear projection of the first 2 PCs of the raw input data space for the "steel" dataset **(left)**, the manifold space using the encoder from [223] **(middle)**, and the augmented manifold space after using my proposed method **(right)**. The colour of the data points depicts class label.

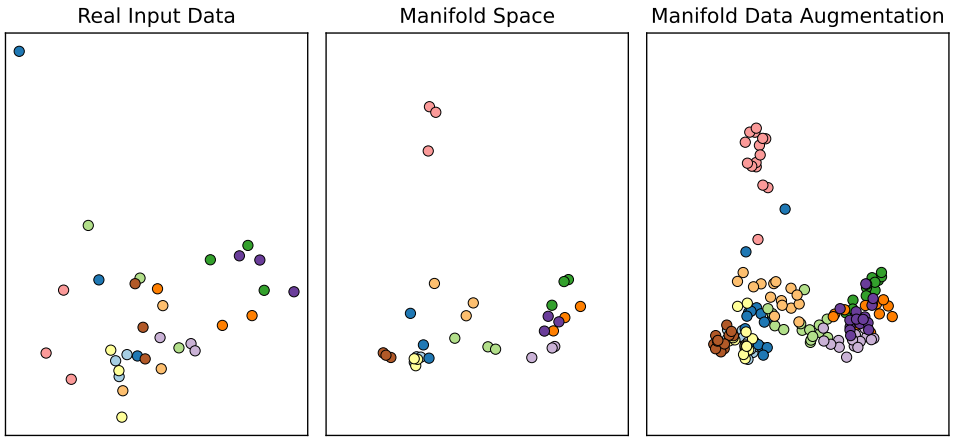

Figure 5.7: PCA linear projection of the first 2 PCs of the raw input data space for the "texture" dataset **(left)**, the manifold space using the encoder from [223] **(middle)**, and the augmented manifold space after using my proposed method **(right)**. The colour of the data points depicts class label.

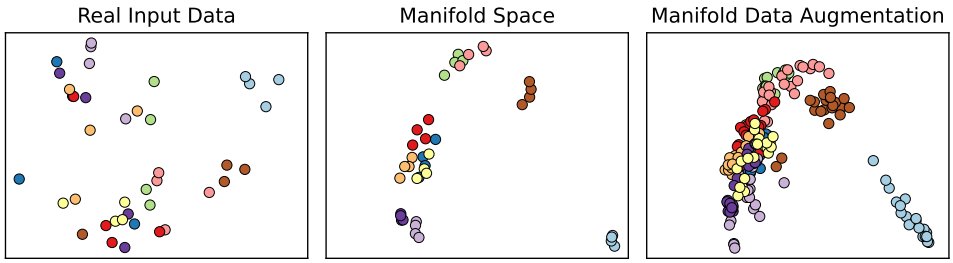

Figure 5.8: PCA linear projection of the first 2 PCs of the raw input data space for the "fourier" dataset **(left)**, the manifold space using the encoder from [223] **(middle)**, and the augmented manifold space after using my proposed method **(right)**. The colour of the data points depicts class label.

5.6 Discussion and summary

In this chapter, I presented TabMDA, a novel training-free manifold data augmentation method specifically designed for tabular data. By leveraging the in-context learning capabilities of a pre-trained model, namely TabPFN, and introducing my innovative in-context subsetting (ICS) mechanism for label-invariant transformations, TabMDA generates diverse and informative augmented representations directly in the learned manifold space. This approach effectively addresses the challenges of augmenting tabular data, where traditional augmentation techniques often fail due to the heterogeneous nature and lack of explicit symmetries in the input space.

My extensive experiments across a variety of small-to-medium sized tabular datasets demonstrate that TabMDA significantly improves the performance of standard classifiers, boosting accuracy by up to 15% in some cases, while simultaneously reducing performance variability across different models. Notably, even simple and interpretable classifiers like KNN benefit from my method, achieving competitive results that typically require more sophisticated ensemble methods. The consistency in performance provided by TabMDA is particularly valuable in real-world scenarios where robustness and reliability are as important as raw predictive accuracy.

A key insight underlying TabMDA is its ability to exploit the manifold learned by pre-trained in-context models. By embedding the data into a structured, continuous space, my method enables meaningful label-invariant transformations through ICS. This not only increases the effective training set size but also helps mitigate overfitting, especially in low-data regimes. The process of generating multiple augmented versions of each data point through different contextual encodings enriches the feature diversity and allows downstream classifiers to generalise better.

Despite these promising results, TabMDA inherits limitations from its underlying in-context model encoder. Specifically, the current implementation using TabPFN is constrained to tabular datasets with up to 100 features. As advancements occur in tabular in-context models, I anticipate corresponding improvements in the performance and scalability of TabMDA. Another significant limitation arises from the requirement that the in-context model must have access to the entire training set at inference time. This constraint can be impractical or even impossible in scenarios with strict privacy concerns or limited computational resources. One potential solution to this challenge is knowledge distillation of the underlying encoder, an approach previously demonstrated to retain most of the original model's performance [258]. Further research into knowledge distillation techniques could effectively mitigate this issue.

Future research could also benefit from exploring adaptive strategies for context subsetting, potentially optimising augmentation quality by dynamically tuning context size based on dataset characteristics. Moreover, investigating extensions to handle mixed data types or integrating TabMDA into semi-supervised learning frameworks could significantly broaden its practical applicability. Lastly, deeper investigations into the properties of the embedding manifold, including examining the embedding spaces of tabular in-context models and the effects of

ICS, could provide valuable insights into further improving augmentation effectiveness and downstream performance.

Overall, TabMDA offers a practical and effective solution for improving tabular data classification. By harnessing the strengths of pre-trained in-context models to perform data augmentation directly in the embedding space, my method advances the state of the art in tabular learning and opens new possibilities for robust, scalable, and interpretable data analysis.

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

This thesis has made foundational contributions to the field of multimodal learning and knowledge representation by addressing critical challenges in three distinct yet interconnected areas: text-attributed node classification on hypergraphs, weakly supervised semantic distillation for dense retrieval, and tabular manifold data augmentation. Through the development of novel methodologies, I have advanced the state-of-the-art in each domain and provided valuable insights into the effective integration of diverse data modalities.

My work on text-attributed hypergraphs led to the design of HyperBERT, a framework that combines the strengths of hypergraph neural networks and pre-trained language models. By embedding higher-order structural information directly into the language model's architecture, HyperBERT demonstrates that incorporating hypergraph topology with rich semantic representations can significantly improve node classification performance. This contribution not only sets a new benchmark for text-attributed node classification but also underscores the untapped potential of jointly modeling complex relational structures and unstructured textual data.

In the area of dense retrieval, I introduced a weakly supervised training framework that leverages language model distillation. By transferring the robust semantic reasoning capabilities of large language models to more compact, knowledge-aware models, my approach effectively mitigates the dependence on extensive and expensive annotated data. The success of this framework on benchmarks such as FEVER and FB15k-237 illustrates that carefully designed self-supervised losses can capture nuanced claim–evidence relationships even in low-resource settings. This advancement opens the door to more scalable and accessible claim verification systems, a necessity in today's era of rampant misinformation.

My work into tabular data augmentation culminated in the development of TabMDA, a training-free method that operates in the embedding manifold of pre-trained in-context models. By introducing in-context subsetting to perform label-invariant transformations, TabMDA not only expands the effective size of the training dataset but also significantly enhances the performance and stability of downstream classifiers. The ability to improve even simple and

interpretable models such as KNN highlights the broad applicability and practical impact of the approach, especially in scenarios where data is scarce.

Looking forward, several promising directions emerge from this research. First, scaling these multimodal systems to accommodate larger, more complex datasets remains an important challenge. Extending HyperBERT to dynamically incorporate temporal and evolving hypergraph structures could enable new applications in social network analysis and dynamic recommendation systems. Similarly, adapting the dense retrieval framework to multilingual and cross-modal contexts would further broaden its applicability and impact, especially for global misinformation detection.

In the realm of tabular data augmentation, future work could explore lightweight, parameter-efficient models that enhance usability in resource-constrained environments. Integrating explainability directly into the augmentation process is another compelling avenue, with the potential to increase user trust in critical domains such as healthcare and finance. Furthermore, a deeper exploration of the learned manifold properties may yield insights into more robust and theoretically grounded augmentation strategies.

Beyond these specific extensions, the methodologies developed in this thesis have the potential to drive interdisciplinary applications. They offer a promising foundation for lifelong learning systems capable of integrating new data modalities and adapting to evolving data distributions. Ultimately, this research lays the groundwork for the next generation of intelligent systems, ones that are not only more versatile and robust but also better equipped to operate in complex, multimodal environments that characterize many real-world problems.

In conclusion, the contributions presented in this thesis represent significant strides towards bridging the gap between heterogeneous data modalities. Using structural, textual, and tabular information within deep learning frameworks, I have opened new pathways for innovation in multimodal learning and set the stage for future research that will further enhance the capabilities and impact of artificial intelligence.

BIBLIOGRAPHY

[1] Ashish Vaswani, Noam Shazeer, Niki Par1r, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020.

[4] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hao Ren, Bo Liu, Piotr Dollár, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[5] Rex Ying, Ruining He, Kaifeng Chen, Praphat Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[6] Alphaeus Dmonte, Roland Oruche, Marcos Zampieri, Prasad Calyam, and Isabelle Augenstein. Claim verification in the age of large language models: A survey, 2025.

[7] Mukund Chaudhry Chaudhry, Arman Kazmi, Shashank Jatav, Akhilesh Verma, Vishal Samal, Kristopher Paul, and Ashutosh Modi. Reducing inference time of biomedical NER tasks using multi-task learning. In Md. Shad Akhtar and Tanmoy Chakraborty,

editors, *Proceedings of the 19th International Conference on Natural Language Processing (ICON)*, pages 116–122, New Delhi, India, December 2022. Association for Computational Linguistics.

[8] Vadim Borisov, Adrian Leemann, Florian Huber, Stefan Ilic, and Christin Seifert. Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*, 2021.

[9] Andreas Voskou, Charalambos Christoforou, and Sotirios Chatzis. Transformers with stochastic competition for tabular data modelling, 2024.

[10] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[11] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989.

[12] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, mar 2010.

[13] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In Bernhard Schölkopf, John C. Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 153–160. MIT Press, 2006.

[14] Marc' aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann Cun. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

[15] Genevieve B. Orr and Klaus-Robert Müller, editors. *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*. Springer, 1998.

[16] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington, editors, *AISTATS*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[18] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[22] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [cs]*, December 2014. arXiv: 1409.3215.

[23] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[24] Michael I. Jordan. Chapter 25 - serial order: A parallel distributed processing approach. In John W. Donahoe and Vivian Packard Dorsel, editors, *Neural-Network Models of Cognition*, volume 121 of *Advances in Psychology*, pages 471–495. North-Holland, 1997.

[25] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, 2014.

[26] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2342–2350. JMLR.org, 2015.

[27] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization, 2015.

[28] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks, 2016.

[29] Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization, 2017.

[30] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[31] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks, 2016.

[32] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas

Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, 2018.

[33] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, jul 2017.

[34] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications, 2018.

[35] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, 2005.

[36] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102, 2009.

[37] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

[38] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks, 2018.

[39] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*, May 2016. arXiv: 1409.0473.

[40] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perez Cistac, Tim Rault, Rémi Louf, Morgan Funchath, et al. Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 2020.

[41] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019.

[42] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training, 2018.

[43] OpenAI. GPT-4 Technical Report, 2023. Available at `https://openai.com/research/gpt-4`.

[44] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[45] Mistral AI. Mistral 7B: Open-Source Efficient Models, 2023. Available at `https://www.mistral.ai/`.

[46] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yuan Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[47] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[48] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[49] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32:5753–5763, 2019.

[50] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2019.

[51] Lin Chen, Zhe Zhang, and Wei Ouyang. Multimodal learning with transformers: A survey. *arXiv preprint arXiv:2109.04894*, 2021.

[52] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning*, pages 689–696, 2011.

[53] Neil Hurley and Shane Rickard. Audio-visual speech processing: Lip synchronisation and fusion. *IEEE Signal Processing Magazine*, 27(3):24–33, 2010.

[54] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[55] Jean-Baptiste Alayrac et al. Flamingo: a visual language model for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[56] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023.

[57] Yutai Wang, Peng Ke, Yijin Zheng, Kam-Fai Huang, Yong Jiang, Xiaoyan Zhu, and Minlie Huang. Semantic matching and aggregation network for few-shot intent detection. *Proceedings of EMNLP*, 2020.

[58] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[59] Yikun Xu, Junchen Zhang, Zhao Liu, Xin Wu, and Hui Zhao. Semantic-enhanced explainable recommendation. *Proceedings of WSDM*, 2020.

[60] Tianliang Chen, Zhixiong Sui, Xiaojie Yuan, and Yongjian Wu. Cross-modal retrieval with unsupervised knowledge distillation. *Proceedings of EMNLP*, 2020.

[61] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[62] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[63] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *NeurIPS*, 2019.

[64] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[65] Tianliang Chen, Zhixiong Sui, Xiaojie Yuan, and Yongjian Wu. Learning cross-modal retrieval with noisy labels. *CVPR*, 2020.

[66] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 2017.

[67] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *EMNLP*, 2018.

[68] Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[69] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. *NAACL*, 2018.

[70] Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Gear: Graph-based evidence aggregating and reasoning for fact verification, 2019.

[71] Yixin Nie, Haonan Chen, and Mohit Bansal. Combining fact extraction and verification with neural semantic matching networks. *AAAI*, 2019.

[72] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *NeurIPS*, 2020.

[73] Panupong Pasupat, Yuan Zhao, Xiang Zhang, and Kelvin Guu. Controlled crowdsourcing for high-quality qa-pair generation. *ACL*, 2021.

[74] Song Bai, Feihu Zhang, and Philip H.S. Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.

[75] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. Hypergraph learning: Methods and practices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2548–2566, 2022.

[76] Jian Hou, Marcello Pelillo, and Huaqiang Yuan. Hypergraph matching via game-theoretic hypergraph clustering. *Pattern Recognition*, 125:108526, 2022.

[77] Hanrui Wu, Yuguang Yan, and Michael Kwok-Po Ng. Hypergraph collaborative network on vertices and hyperedges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3245–3258, 2023.

[78] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3558–3565, Jul. 2019.

[79] Jian Hou, Huaqiang Yuan, and Marcello Pelillo. Game-theoretic hypergraph matching with density enhancement. *Pattern Recognition*, 133:109035, 2023.

[80] Hanrui Wu, Jinyi Long, Nuosi Li, Dahai Yu, and Michael K. Ng. Adversarial auto-encoder domain adaptation for cold-start recommendation with positive and negative hypergraphs. *ACM Trans. Inf. Syst.*, 41(2), dec 2022.

[81] Hanrui Wu, Chung Wang Wong, Jia Zhang, Yuguang Yan, Dahai Yu, Jinyi Long, and Michael K. Ng. Cold-start next-item recommendation by user-item matching and auto-encoders. *IEEE Transactions on Services Computing*, 16(4):2477–2489, 2023.

[82] Kazuki Nakajima and Takeaki Uno. Inferring community structure in attributed hypergraphs using stochastic block models, 2024.

[83] Andrea Failla, Salvatore Citraro, and Giulio Rossetti. Attributed stream hypergraphs: temporal modeling of node-attributed high-order interactions. *Applied Network Science*, 8(1), June 2023.

[84] Sebastiano Smaniotto and Marcello Pelillo. Two metrics for attributed hypergraphs. *Pattern Recognition Letters*, 149:143–149, 2021.

[85] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. Graphformers: Gnn-nested transformers for representation learning on textual graph. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 28798–28810. Curran Associates, Inc., 2021.

[86] Martina Contisciani, Eleanor A. Power, and Caterina De Bacco. Community detection with node attributes in multilayer networks. *Scientific Reports*, 10(1), September 2020.

[87] Raj Kumar Pan, Kimmo Kaski, and Santo Fortunato. World citation and collaboration networks: uncovering the role of geography in science. *Scientific Reports*, 2(1), November 2012.

[88] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs, 2021.

[89] Petr Chunaev. Community detection in node-attributed social networks: A survey. *Computer Science Review*, 37:100286, 2020.

[90] Aleix Bassolas, Anton Holmgren, Antoine Marot, Martin Rosvall, and Vincenzo Nicosia. Mapping nonlocal relationships between metadata and network structure with metadata-dependent encoding of random walks. *Science Advances*, 8(43):eabn7558, 2022.

[91] M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

[92] Alice Patania, Giovanni Petri, and Francesco Vaccarino. The shape of collaborations. *EPJ Data Science*, 6(1), August 2017.

[93] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. SPECTER: Document-level representation learning using citation-informed transformers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online, July 2020. Association for Computational Linguistics.

[94] Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. Textgnn: Improving text encoder via graph neural network in sponsored search. In *Proceedings of the Web Conference 2021*, WWW '21, page 2848–2857, New York, NY, USA, 2021. Association for Computing Machinery.

[95] Jose Lugo-Martinez, Daniel Zeiberg, Thomas Gaudelet, Noël Malod-Dognin, Natasa Przulj, and Predrag Radivojac. Classification in biological networks with hypergraphlet kernels. *Bioinformatics*, 37(7):1000–1007, 09 2020.

[96] Yiran Li, Renchi Yang, and Jieming Shi. Efficient and effective attributed hypergraph clustering via k-nearest neighbor augmentation. *Proc. ACM Manag. Data*, 1(2), jun 2023.

[97] Longcan Wu, Daling Wang, Kaisong Song, Shi Feng, Yifei Zhang, and Ge Yu. Dual-view hypergraph neural networks for attributed graph learning. *Knowledge-Based Systems*, 227:107185, 2021.

[98] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[99] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, July 2020. arXiv:1910.10683 [cs, stat].

[100] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[101] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[102] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations*, 2023.

[103] Haoyu Kuang, Jiarong Xu, Haozhe Zhang, Zuyu Zhao, Qi Zhang, Xuanjing Huang, and Zhongyu Wei. Unleashing the power of language models in text-attributed graph. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8429–8441, Singapore, December 2023. Association for Computational Linguistics.

[104] Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. Node classification in social networks. *ArXiv*, abs/1101.3291, 2011.

[105] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5171–5181, Red Hook, NY, USA, 2018. Curran Associates Inc.

[106] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[107] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[108] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[109] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1–4):43–52, August 2010.

[110] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[111] Hanrui Wu and Michael K. Ng. Hypergraph convolution on nodes-hyperedges network for semi-supervised node classification. *ACM Trans. Knowl. Discov. Data*, 16(4), jan 2022.

[112] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach. In *The World Wide Web Conference*, WWW '19, page 2147–2157, New York, NY, USA, 2019. Association for Computing Machinery.

[113] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. Next-item recommendation with sequential hypergraphs. In *SIGIR*, 2020.

[114] Zizhao Zhang, Haojie Lin, Xibin Zhao, Rongrong Ji, and Yue Gao. Inductive multi-hypergraph learning and its application on view-based 3d object classification. *IEEE Transactions on Image Processing*, 27(12):5957–5968, 2018.

[115] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, pages 1509–1520. Curran Associates, Inc., 2019.

[116] Yifan Feng, Haoxuan You, Zhirong Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3558–3565, 2019.

[117] Sunwoo Kim, Shinhwan Kang, Fanchen Bu, Soo Yong Lee, Jaemin Yoo, and Kijung Shin. Hypeboy: Generative self-supervised representation learning on hypergraphs. In *The Twelfth International Conference on Learning Representations*, 2024.

[118] Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. Greaselm: Graph reasoning enhanced language models. In *International Conference on Learning Representations*, 2021.

[119] Bowen Jin, Yu Zhang, Yu Meng, and Jiawei Han. Edgeformers: Graph-empowered transformers for representation learning on textual-edge networks. In *International Conference on Learning Representations*, 2023.

[120] Bowen Jin, Yu Zhang, Qi Zhu, and Jiawei Han. Heterformer: Transformer-based deep node representation learning on heterogeneous text-rich networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1020–1031, 2023.

[121] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S. Dhillon. Node feature extraction by self-supervised multi-scale neighborhood prediction. *ArXiv*, abs/2111.00064, 2021.

[122] Jiong Zhang, Wei-Cheng Chang, Hsiang-Fu Yu, and Inderjit S Dhillon. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[123] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning, 2023.

[124] Vassilis N. Ioannidis, Xiang Song, Da Zheng, Houyu Zhang, Jun Ma, Yi Xu, Belinda Zeng, Trishul Chilimbi, and George Karypis. Efficient and effective training of language and graph neural network models, 2022.

[125] Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. Simteg: A frustratingly simple approach improves textual graph learning, 2023.

[126] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-bert: Enabling language representation with knowledge graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2901–2908, Apr. 2020.

[127] Liang Yao, Chengsheng Mao, and Yuan Luo. Kg-bert: Bert for knowledge graph completion, 2019.

[128] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017. arXiv: 1706.03762.

[129] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models, 2013.

[130] Yihe Dong, Will Sawin, and Yoshua Bengio. Hnhn: Hypergraph networks with hyperedge neurons. *ICML Graph Representation Learning and Beyond Workshop*, 2020.

[131] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2021.

[132] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations*, 2022.

[133] Tianxin Wei, Yuning You, Tianlong Chen, Yang Shen, Jingrui He, and Zhangyang Wang. Augmentations in hypergraph contrastive learning: Fabricated and generative. *arXiv preprint arXiv:2210.03801*, 2022.

[134] Yizhen Zheng, Shirui Pan, Vincent Lee, Yu Zheng, and Philip S. Yu. Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[135] Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. Equivariant hypergraph diffusion neural operators. In *The Eleventh International Conference on Learning Representations*, 2023.

[136] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, December 2019. arXiv:1912.01703 [cs, stat].

[137] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[138] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric, April 2019. arXiv:1903.02428 [cs, stat].

[139] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[140] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators, 2020.

[141] Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. A Survey on Automated Fact-Checking. *Transactions of the Association for Computational Linguistics*, 10:178–206, 02 2022.

[142] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and verification. *CoRR*, abs/1803.05355, 2018.

[143] Arkaitz Zubiaga, Ahmet Aker, Kalina Bontcheva, Maria Liakata, and Rob Procter. Detection and resolution of rumours in social media: A survey. *ACM Comput. Surv.*, 51(2), feb 2018.

[144] Juraj Vladika and Florian Matthes. Scientific fact-checking: A survey of resources and approaches. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[145] Anubrata Das, Houjiang Liu, Venelin Kovatchev, and Matthew Lease. The state of human-centered NLP technology for fact-checking. *Information Processing & Management*, 60(2):103219, 2023.

[146] Xiaowei Huang, Wenjie Ruan, Wei Huang, Gaojie Jin, Yi Dong, Changshun Wu, Saddek Bensalem, Ronghui Mu, Yi Qi, Xingyu Zhao, Kaiwen Cai, Yanghao Zhang, Sihao Wu, Peipei Xu, Dengyu Wu, Andre Freitas, and Mustafa A. Mustafa. A survey of safety and trustworthiness of large language models through the lens of verification and validation, 2023.

[147] Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. Trustworthy llms: a survey and guideline for evaluating large language models' alignment, 2023.

[148] Isabelle Augenstein, Christina Lioma, Dongsheng Wang, Lucas Chaves Lima, Casper Hansen, Christian Hansen, and Jakob Grue Simonsen. Multifc: A real-world multi-domain dataset for evidence-based fact checking of claims. *CoRR*, abs/1909.03242, 2019.

[149] Jiasheng Si, Deyu Zhou, Tongzhe Li, Xingyu Shi, and Yulan He. Topic-aware evidence reasoning and stance-aware aggregation for fact verification, 2021.

[150] Biru Zhu, Xingyao Zhang, Ming Gu, and Yangdong Deng. Knowledge enhanced fact checking and verification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3132–3143, 2021.

[151] Jackson Luken, Nanjiang Jiang, and Marie-Catherine de Marneffe. QED: A fact verification system for the FEVER shared task. In James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal, editors, *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 156–160, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[152] Wenpeng Yin and Dan Roth. Twowingos: A two-wing optimization strategy for evidential claim verification, 2018.

[153] Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. Coreferential reasoning learning for language representation, 2020.

[154] Wanjun Zhong, Jingjing Xu, Duyu Tang, Zenan Xu, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. Reasoning over semantic-level graph for fact checking, 2020.

[155] Chonghao Chen, Fei Cai, Xuejun Hu, Wanyu Chen, and Honghui Chen. Hhgn: A hierarchical reasoning-based heterogeneous graph neural network for fact verification. *Inf. Process. Manage.*, 58(5), sep 2021.

[156] Chonghao Chen, Fei Cai, Xuejun Hu, Jianming Zheng, Yanxiang Ling, and Honghui Chen. An entity-graph based reasoning method for fact verification. *Information Processing & Management*, 58(3):102472, 2021.

[157] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. Fine-grained fact verification with kernel graph attention network, 2021.

[158] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[159] Weijie Chen, Shiliang Pu, Di Xie, Shicai Yang, Yilu Guo, and Luojun Lin. Unsupervised image classification for deep representation learning, 2020.

[160] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc., 2020.

[161] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020.

[162] Mayank Jobanputra. Unsupervised question answering for fact-checking. In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 52–56, Hong Kong, China, November 2019. Association for Computational Linguistics.

[163] Jiseong Kim and Key-sun Choi. Unsupervised fact checking by counter-weighted positive and negative evidential paths in a knowledge graph. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1677–1686, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[164] Shailza Jolly, Pepa Atanasova, and Isabelle Augenstein. Generating fluent fact checking explanations with unsupervised post-editing. *Information*, 13(10), 2022.

[165] Fengzhu Zeng and Wei Gao. Prompt to be consistent is better than self-consistent? few-shot and zero-shot fact verification with pre-trained language models. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[166] Yu Chen, Lingfei Wu, and Mohammed J Zaki. Bidirectional attentive memory networks for question answering over knowledge bases. *arXiv preprint arXiv:1903.02188*, 2019.

[167] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings*

of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4498–4507, 2020.

[168] Nayeon Lee, Belinda Z. Li, Sinong Wang, Wen-tau Yih, Hao Ma, and Madian Khabsa. Language models as fact checkers? In *Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER)*, pages 36–41, Online, July 2020. Association for Computational Linguistics.

[169] Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. Generate rather than retrieve: Large language models are strong context generators. In *International Conference for Learning Representation (ICLR)*, 2023.

[170] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[171] Chonghao Chen, Jianming Zheng, and Honghui Chen. Cosg: A graph-based contrastive learning method for fact verification. *Sensors*, 21(10), 2021.

[172] Michael Mu, Sreyasee Das Bhattacharjee, and Junsong Yuan. Self-supervised distilled learning for multi-modal misinformation identification. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2818–2827, 2023.

[173] Chonghao Chen, Jianming Zheng, and Honghui Chen. Knowledge-enhanced graph attention network for fact verification. *Mathematics*, 9(16), 2021.

[174] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[175] Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. LaPraDoR: Unsupervised pretrained dense retriever for zero-shot text retrieval. In *ACL 2022 (Findings)*, 2022.

[176] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[177] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015.

[178] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation, 2019.

[179] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, 2017.

[180] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation, 2022.

[181] Guodong Xu, Ziwei Liu, Xiaoxiao Li, and Chen Change Loy. Knowledge distillation meets self-supervision, 2020.

[182] Zhiyuan Fang, Jianfeng Wang, Lijuan Wang, Lei Zhang, Yezhou Yang, and Zicheng Liu. Seed: Self-supervised distillation for visual representation, 2021.

[183] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), jan 2020.

[184] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y Hammerla. Relational graph attention networks, 2019.

[185] Jun Xia, Lirong Wu, Ge Wang, and Stan Z. Li. Progcl: Rethinking hard negative mining in graph contrastive learning. In *International conference on machine learning*. PMLR, 2022.

[186] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. Knowledge graph contrastive learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 1434–1443, New York, NY, USA, 2022. Association for Computing Machinery.

[187] Md Rashad Al Hasan Rony, Mirza Mohtashim Alam, Semab Ali, Jens Lehmann, and Sahar Vahdati. Lemon: Language model for negative sampling of knowledge graph embeddings, 2022.

[188] Yinquan Lu, Haonan Lu, Guirong Fu, and Qun Liu. Kelm: Knowledge enhanced pre-trained language representations with message passing on hierarchical relational graphs, 2022.

[189] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and improving representations with the soft nearest neighbor loss, 2019.

[190] Wei Chen, Tie-yan Liu, Yanyan Lan, Zhi-ming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[191] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler: A unified model for knowledge embedding and pre-trained

language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021.

[192] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[193] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2023.

[194] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

[195] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[196] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[197] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Unsupervised semantic segmentation by contrasting object mask proposals, 2021.

[198] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning, 2020.

[199] Luca Di Liello, Siddhant Garg, Luca Soldaini, and Alessandro Moschitti. Paragraph-based transformer pre-training for multi-sentence inference. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2521–2531, Seattle, United States, July 2022. Association for Computational Linguistics.

[200] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. Gere: Generative evidence retrieval for fact verification. In *Proceedings of the 45th International ACM*

SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, page 2184–2189, New York, NY, USA, 2022. Association for Computing Machinery.

[201] Amrith Krishna, Sebastian Riedel, and Andreas Vlachos. Proofver: Natural logic theorem proving for fact verification. *CoRR*, abs/2108.11357, 2021.

[202] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[203] Jiabang He, Liu Jia, Lei Wang, Xiyao Li, and Xing Xu. Mocosa: Momentum contrast for knowledge graph completion with structure-augmented pre-trained language models, 2023.

[204] Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021*. ACM, apr 2021.

[205] Chen Chen, Yufei Wang, Bing Li, and Kwok-Yan Lam. Knowledge is flat: A Seq2Seq generative framework for various knowledge graph completion. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na, editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4005–4017, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

[206] Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models, 2022.

[207] Lisiane B Meira, Antonio MC Reis, David L Cheo, Dorit Nahari, Dennis K Burns, and Errol C Friedberg. Cancer predisposition in mutant mice defective in multiple genetic pathways: uncovering important genetic interactions. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis*, 477(1-2):51–58, 2001.

[208] Rubika Balendra and Adrian M Isaacs. C9orf72-mediated als and ftd: multiple pathways to disease. *Nature Reviews Neurology*, 14(9):544–558, 2018.

[209] Matthew Kelly and Christopher Semsarian. Multiple mutations in genetic cardiovascular disease: a marker of disease severity? *Circulation: Cardiovascular Genetics*, 2(2):182–190, 2009.

[210] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):4308, 2014.

[211] Gregor Kasieczka, Benjamin Nachman, David Shih, Oz Amram, Anders Andreassen, Kees Benkendorfer, Blaz Bortolato, Gustaaf Brooijmans, Florencia Canelli, Jack H Collins, et al. The lhc olympics 2020 a community challenge for anomaly detection in high energy physics. *Reports on progress in physics*, 84(12):124201, 2021.

[212] Zenan Zhai, Christian Druckenbrodt, Camilo Thorne, Saber A Akhondi, Dat Quoc Nguyen, Trevor Cohn, and Karin Verspoor. Chemtables: a dataset for semantic classification on tables in chemical patents. *Journal of Cheminformatics*, 13(1):1–20, 2021.

[213] John A Keith, Valentin Vassilev-Galindo, Bingqing Cheng, Stefan Chmiela, Michael Gastegger, Klaus-Robert Muller, and Alexandre Tkatchenko. Combining machine learning and computational chemistry for predictive insights into chemical systems. *Chemical reviews*, 121(16):9816–9872, 2021.

[214] Andrei Margeloiu, Nikola Simidjievski, Pietro Lio, and Mateja Jamnik. Weight predictor network with feature selection for small sample tabular biomedical data. *AAAI Conference on Artificial Intelligence*, 2023.

[215] Xiangjian Jiang, Andrei Margeloiu, Nikola Simidjievski, and Mateja Jamnik. Protogate: Prototype-based neural networks with global-to-local feature selection for tabular biomedical data. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[216] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 2019.

[217] Soma Onishi and Shoya Meguro. Rethinking data augmentation for tabular data in deep learning, 2023.

[218] Dionysis Manousakas and Sergül Aydöre. On the usefulness of synthetic tabular data generation. *Data-centric Machine Learning Research (DMLR) Workshop at the 40th International Conference on Machine Learning (ICML)*, 2023.

[219] Yunrui Sheng and Liang Xiao. Manifold augmentation based self-supervised contrastive learning for few-shot remote sensing scene classification. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 2239–2242, 2022.

[220] Shulei Wang. Augmentation invariant manifold learning. *arXiv preprint arXiv:2211.00460*, 2023.

[221] Joao Fonseca and Fernando Bacao. Tabular and latent space synthetic data generation: a literature review. *Journal of Big Data*, 10(1), 2023.

[222] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.

[223] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[224] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[225] Wei-Yao Wang, Wei-Wei Du, Derek Xu, Wei Wang, and Wen-Chih Peng. A survey on self-supervised learning for non-sequential tabular data, 2024.

[226] Chao Ye, Guoshan Lu, Haobo Wang, Liyao Li, Sai Wu, Gang Chen, and Junbo Zhao. Towards cross-table masked pretraining for web data mining. In *Proceedings of the ACM on Web Conference 2024*, pages 4449–4459, 2024.

[227] Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. Lift: Language-interfaced fine-tuning for non-language machine learning tasks. In *Advances in Neural Information Processing Systems*, volume 35, pages 11763–11784, 2022.

[228] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

[229] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 5549–5581, 2023.

[230] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang,

Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations (ICLR)*, 2022.

[231] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. Tuta: Tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790, 2021.

[232] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4320–4333. Association for Computational Linguistics, 2020.

[233] Zifeng Wang and Jimeng Sun. Transtab: Learning transferable tabular transformers across tables. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[234] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. In *Table Representation Learning Workshop at NeurIPS*, 2022.

[235] Sercan O Arık and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 6679–6687, 2021.

[236] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings, 2020.

[237] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *NeurIPS*, 2021.

[238] Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

[239] Andrei Margeloiu, Nikola Simidjievski, Pietro Lio, and Mateja Jamnik. Gcondnet: A novel method for improving neural networks on small high-dimensional tabular data. In *2nd Table Representation Learning Workshop at NeurIPS 2023*, 2023.

[240] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations (ICLR)*, 2022.

[241] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[242] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019.

[243] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

[244] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, pages 17564–17579. PMLR, 2023.

[245] David S Watson, Kristin Blesch, Jan Kapar, and Marvin N Wright. Adversarial random forests for density estimation and generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 5357–5375. PMLR, 2023.

[246] Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. Goggle: Generative modelling for tabular data by learning relational structure. In *The Eleventh International Conference on Learning Representations*, 2023.

[247] Georgios Douzas and Fernando Bacao. Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with applications*, 91:464–471, 2018.

[248] Yiming Qin, Huangjie Zheng, Jiangchao Yao, Mingyuan Zhou, and Ya Zhang. Class-balancing diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18434–18443, 2023.

[249] Vignesh Sampath, Iñaki Maurtua, Juan Jose Aguilar Martin, and Aitor Gutierrez. A survey on generative adversarial networks for imbalance problems in computer vision tasks. *Journal of big Data*, 8:1–59, 2021.

[250] Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. Tabpfgen–tabular data generation with tabpfn. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

[251] Nabeel Seedat, Nicolas Huynh, Boris van Breugel, and Mihaela van der Schaar. Curated llm: Synergy of llms and data curation for tabular augmentation in ultra low-data regimes. *arXiv preprint arXiv:2312.12112*, 2023.

[252] B. Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS 2021)*, 2021.

[253] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[254] L. Breiman, Jerome H. Friedman, Richard A. Olshen, and C. J. Stone. Classification and regression trees. *Biometrics*, 40:874, 1984.

[255] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[256] Boris Van Breugel, Zhaozhi Qian, and Mihaela Van Der Schaar. Synthetic data, real errors: how (not) to publish and use synthetic data. In *International Conference on Machine Learning*, pages 34793–34808. PMLR, 2023.

[257] Samuel G Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 774–782, 2021.

[258] Andreas Müller, Carlo Curino, and Raghu Ramakrishnan. Mothernet: A foundational hypernetwork for tabular classification. *arXiv preprint arXiv:2312.08598*, 2023.